



Ping Identity[®] Directory Server Administration Guide

Version 6.0.1.0

Copyright

This document constitutes an unpublished, copyrighted work and contains valuable trade secrets and other confidential information belonging to UnboundID Corporation. None of the foregoing material may be copied, duplicated, or disclosed to third parties without the express written permission of UnboundID Corporation.

This distribution may include materials developed by third parties. Third-party URLs are also referenced in this document. UnboundID is not responsible for the availability of third-party web sites mentioned in this document. UnboundID does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. UnboundID will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

“UnboundID” is a registered trademark of UnboundID Corporation. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

The contents of this publication are presented for information purposes only and are provided “as is”. While every effort has been made to ensure the accuracy of the contents, the contents are not to be construed as warranties or guarantees, expressed or implied, regarding the products or services described herein or their use or applicability. We reserve the right to modify or improve the design or specifications of such products at any time without notice.

Copyright 2017 UnboundID Corporation

All Rights Reserved

Published: 2017-02-13

Contents

- Preface..... xix**
 - Purpose of This Guide..... xix
 - Audience..... xix
 - Related Documentation..... xix
 - Document Conventions..... xx

- Chapter 1: Overview of the Server..... 1**
 - Server Features..... 2
 - Administration Framework..... 3
 - Server Tools Location..... 4

- Chapter 2: Preparing Your Environment..... 5**
 - Before You Begin..... 6
 - Installing Java..... 6
 - Preparing the Operating System (Solaris)..... 6
 - Using ZFS..... 7
 - To Restrict ZFS Memory Consumption..... 7
 - To Limit ZFS Transaction Group Writes..... 8
 - ZFS Access to Underlying Disks..... 8
 - Configuring ZFS Compression..... 9
 - To Disable the Access Time Update for Reads..... 9
 - Preparing the Operating System (Linux)..... 10
 - Configuring the File Descriptor Limits..... 10
 - File System Tuning..... 11
 - Setting the Maximum User Processes..... 12
 - About Editing OS-Level Environment Variables..... 12
 - Install sysstat and pstack (Red Hat)..... 12
 - Install dstat (SUSE Linux)..... 13
 - Disable Filesystem Swapping..... 13
 - Omit vm.overcommit_memory..... 13
 - Managing System Entropy..... 13
 - Running as a Non-Root User..... 14
 - Running as a Non-Root User (Linux)..... 14
 - Enabling the Server to Listen on Privileged Ports (Linux)..... 15
 - Creating a Solaris Role..... 15

- Chapter 3: Installing the Server..... 17**
 - Getting the Installation Packages..... 18
 - To Unpack the Build Distribution..... 18
 - About the RPM Package..... 18
 - About the Layout of the Directory Server Folders..... 19
 - About the Server Installation Modes..... 19
 - Before You Begin..... 20
 - Setting Up the Directory Server in Interactive Mode..... 21
 - To Install the Directory Server in Interactive Mode..... 21
 - Installing the Directory Server in Non-Interactive Mode..... 23
 - To Install the Directory Server in Non-Interactive Mode..... 23

To Install the Directory Server in Non-Interactive Mode with a Truststore.....	23
Installing a Lightweight Server.....	24
Running the Status Tool.....	25
To Run the Status Tool.....	25
Where To Go From Here.....	26
Working with Multiple Backends.....	27
Importing Data.....	27
Generating Sample Data.....	27
To Import Data on the Directory Server Using Offline Import.....	28
Running the Server.....	28
To Start the Directory Server.....	29
To Run the Server as a Foreground Process.....	29
To Start the Server at Boot Time.....	29
Logging into the Administrative Console.....	30
Stopping the Directory Server.....	30
To Stop the Server.....	30
To Schedule a Server Shutdown.....	30
To Restart the Server.....	31
Uninstalling the Server.....	31
To Uninstall the Server in Interactive Mode.....	32
To Uninstall the Server in Non-Interactive Mode.....	33
To Uninstall Selected Components in Non-Interactive Mode.....	33

Chapter 4: Upgrading the Server..... 35

Upgrade Overview and Considerations.....	36
To Upgrade the Directory Server.....	36
To Upgrade the RPM Package.....	37
Reverting an Update.....	37
Configure SCIM After Upgrade.....	38

Chapter 5: Tuning the Server..... 39

About Minimizing Disk Access.....	40
Memory Allocation and Database Cache.....	40
Directory Server Process Memory.....	40
A Method for Determining Heap and Database Cache Size.....	41
Automatic DB Cache Percentages.....	42
Automatic Memory Allocation.....	42
Automatic Memory Allocation for the Command-Line Tools.....	43
Database Preloading.....	43
Configuring Database Preloading.....	44
Databases on Storage Area Networks, Network-Attached Storage, or running in Virtualized Environment.....	45
Database Cleaner.....	46
Compacting Common Parent DNs.....	47
Import Thread Count.....	47
JVM Properties for Server and Command-Line Tools.....	47
Applying Changes Using dsjavaproperties.....	48
JVM Garbage Collection Using CMS.....	49
To Determine the CMSInitiatingOccupanyFraction.....	50
Tuning For Disk-Bound Deployments.....	51
To Tune for Disk-Bound Deployments.....	52
Uncached Attributes and Entries.....	52
To Configure Uncached Attributes and Entries.....	54

Chapter 6: Configuring the Server..... 55

- Accessing the Directory Server Configuration..... 56
- About dsconfig Configuration Tool..... 56
 - Using dsconfig in Interactive Command-Line Mode..... 56
 - Using dsconfig in Non-Interactive Mode..... 59
 - Getting the Equivalent dsconfig Non-Interactive Mode Command..... 61
 - Using dsconfig Batch Mode..... 61
- Using the Configuration API..... 62
 - Authentication and Authorization with the Configuration API..... 63
 - Relationship Between the Configuration API and the dsconfig Tool..... 63
 - GET Example..... 64
 - GET List Example..... 65
 - PATCH Example..... 66
 - Configuration API Paths..... 69
 - Sorting and Filtering Objects..... 70
 - Updating Properties..... 71
 - Administrative Actions..... 72
 - Updating Servers and Server Groups..... 73
 - Configuration API Responses..... 73
- Configure the Server Using the Administrative Console..... 74
 - To Log onto the Administrative Console..... 74
 - To Configure the Server Using the Console..... 75
- Generating a Summary of Configuration Components..... 78
 - To Generate a Summary of Configuration Components..... 78
- About Root User, Administrator, and Global Administrators..... 78
- Managing Root Users Accounts..... 79
- Default Root Privileges..... 79
- Configuring Administrator Accounts..... 81
 - To Set Up a Single Administrator Account..... 81
 - To Change the Administrator Password..... 83
 - To Set Up an Administrator Group..... 83
- Configuring a Global Administrator..... 84
 - To Create a Global Administrator..... 85
 - To Remove a Global Administrator..... 85
- Configuring Server Groups..... 85
 - About the Server Group Example..... 86
 - To Create a Server Group..... 86
- Configuring Client Connection Policies..... 87
 - Understanding the Client Connection Policy..... 88
 - When a Client Connection Policy is Assigned..... 88
 - Restricting the Type of Search Filter Used by Clients..... 89
 - Setting Resource Limits..... 90
 - Defining the Operation Rate..... 93
 - Client Connection Policy Deployment Example..... 93
- Securing the Server with Lockdown Mode..... 98
 - To Manually Enter Lockdown Mode..... 98
 - To Leave Lockdown Mode..... 99
 - To Start a Server in Lockdown Mode..... 99
- Configuring Maximum Shutdown Time..... 99
 - To Configure the Maximum Shutdown Time..... 99
- Working with Referrals..... 99
 - Specifying LDAP URLs..... 100
 - Creating Referrals..... 101
 - To Modify a Referral..... 101

To Delete a Referral.....	101
Configuring a Read-Only Server.....	102
To Configure a Read-Only Server.....	102
Configuring HTTP Access for the Directory Server.....	103
Configuring HTTP Servlet Extensions.....	103
Web Application Servlet Extensions.....	103
Java-based Servlet Extensions.....	104
Groovy-Scripted Extensions.....	104
Configuring HTTP Operation Loggers.....	104
Example HTTP Log Publishers.....	105
Configuring HTTP Connection Handlers.....	107
Domain Name Service (DNS) Caching.....	109
IP Address Reverse Name Lookups.....	109
Configuring Traffic Through a Load Balancer.....	110
Working with the Referential Integrity Plug-in.....	110
To Enable the Referential Integrity Plug-in.....	111
Working with the Unique Attribute Plug-in.....	111
To Enable the Unique Attribute Plug-in.....	111
Configuring Uniqueness Across Attribute Sets.....	112
To Enable Uniqueness Across Attribute Sets.....	113
Working with the Last Access Time Plug-In.....	113
Working with the Pass Through Authentication Plug-In.....	114
Supporting Unindexed Search Requests.....	115
Sun/Oracle Compatibility.....	116
To Configure the Directory Server for Sun/Oracle Compatibility.....	116

Chapter 7: Configuring Soft Deletes..... 119

About Soft Deletes.....	120
General Tips on Soft Deletes.....	121
Configuring Soft Deletes on the Server.....	123
Configuring Soft Deletes as a Global Configuration.....	123
Searching for Soft Deletes.....	125
To Run a Base-Level Search on a Soft-Deleted Entry.....	126
To Run a Filtered Search by soft-delete-entry Object Class.....	126
To Run a Search using the Soft Delete Entry Access Control.....	127
Undeleting a Soft-Deleted Entry Using the Same RDN.....	128
To Undelete a Soft-Deleted Entry Using the Same RDN.....	128
Undeleting a Soft-Deleted Entry Using a New RDN.....	128
To Undelete a Soft-Deleted-Entry Using a New RDN.....	128
Modifying a Soft-Deleted Entry.....	129
To Modify a Soft-Deleted Entry.....	129
Hard Deleting a Soft-Deleted Entry.....	130
To Hard Delete a Soft-Deleted Entry (Global Configuration).....	130
To Hard Delete a Soft-Deleted Entry (Connection or Request Criteria).....	130
Disabling Soft Deletes as a Global Configuration.....	130
To Disable Soft Deletes as a Global Configuration.....	131
Configuring Soft Deletes by Connection Criteria.....	131
To Enable Soft Deletes by Connection Criteria.....	131
To Disable Soft Deletes by Connection Criteria.....	132
Configuring Soft Deletes by Request Criteria.....	132
To Enable Soft Deletes by Request Criteria.....	132
To Disable Soft Deletes by Request Criteria.....	132
Configuring Soft Delete Automatic Purging.....	133
To Configure Soft-Delete Automatic Purging.....	133
To Disable Soft-Delete Automatic Purging.....	134

Summary of Soft and Hard Delete Processed.....134
 Summary of Soft Delete Controls and Tool Options..... 135
 Monitoring Soft Deletes..... 136
 New Monitor Entries..... 136
 Access Logs..... 137
 Audit Logs..... 138
 Change Log..... 139

Chapter 8: Importing and Exporting Data..... 141

Importing Data..... 142
 Validating an LDIF File..... 142
 To Validate an LDIF File.....142
 Computing Database Cache Estimate..... 143
 Tracking Skipped and Rejected Entries..... 143
 Running an Offline Import..... 143
 To Perform an Offline Import..... 143
 To Perform an Offline LDIF Import Using a Compressed File..... 143
 To Perform an Offline LDIF Import Using a MakeLDIF Template..... 144
 Running an Online LDIF Import..... 144
 To Perform an Online LDIF Import..... 144
 To Schedule an Online Import..... 144
 To Cancel a Scheduled Import..... 145
 Adding Entries to an Existing Directory Server..... 145
 To Append Entries to an Existing Directory Server..... 146
 Filtering Data Import..... 146
 Exporting Data..... 146
 To Perform an Export..... 147
 To Perform an Export from Specific Branches..... 147
 Encrypting LDIF Exports and Signing LDIF Files..... 147
 To Encrypt an LDIF Export..... 147
 To Import an Encrypted LDIF File..... 148
 To Sign an Export..... 148
 To Import a Signed LDIF File..... 148
 Filtering Data Exports..... 148
 Scrambling Data Files..... 149

Chapter 9: Backing Up and Restoring Data..... 153

Backing Up and Restoring Data..... 154
 To List the Available Backups on the System.....154
 To Back Up All Backends..... 155
 To Back Up a Single Backend..... 155
 To Perform an Offline Restore..... 155
 To Assign an ID to a Backup..... 155
 To Run an Incremental Backup on All Backends..... 156
 To Run an Incremental Backup on a Single Backend..... 156
 To Run an Incremental Backup based on a Specific Prior Backup..... 156
 To Restore an Incremental Backup..... 156
 To Schedule an Online Backup..... 156
 To Schedule an Online Restore..... 157
 To Encrypt a Backup..... 157
 To Sign a Hash of the Backup..... 157
 To Restore a Backup..... 158
 Moving or Restoring a User Database..... 158

Comparing the Data in Two Directory Servers.....	158
To Compare Two Directory Servers Using ldap-diff.....	159
To Compare Configuration Entries Using ldap-diff.....	160
To Compare Entries Using Source and Target DN Files.....	161
To Compare Directory Servers for Missing Entries Only Using ldap-diff.....	161

Chapter 10: Working with Indexes..... 163

Overview of Indexes.....	164
General Tips on Indexes.....	164
Index Types.....	166
System Indexes.....	166
To View the System Indexes.....	167
Managing Local DB Indexes.....	167
To View the List of Local DB Indexes.....	167
To View a Property for All Local DB Indexes.....	168
To View the Configuration Parameters for Local DB Index.....	168
To Modify the Configuration of a Local DB Index.....	168
To Create a New Local DB Index.....	169
To Delete a Local DB Index.....	169
Working with JSON Indexes.....	169
Working with Local DB VLV Indexes.....	171
To View the List of Local DB VLV Indexes.....	171
To Create a New Local DB VLV Index.....	171
To Modify a VLV Index’s Configuration.....	171
To Delete a VLV Index.....	172
Working with Filtered Indexes.....	172
To Create a Filtered Index.....	173
Tuning Indexes.....	173
About the Exploded Index Format.....	174
Monitoring Index Entry Limits.....	175
About the Index Summary Statistics Table.....	176
About the dbtest Index Status Table.....	177
Configuring the Index Properties.....	177

Chapter 11: Managing Entries..... 179

Searching Entries.....	180
To Search the Root DSE.....	180
To Search All Entries in the Directory Server.....	181
To Search for an Access Control Instruction.....	181
To Search for the Schema.....	181
To Search for a Single Entry using Base Scope and Base DN.....	181
To Search for a Single Entry Using the Search Filter.....	181
To Search for All Immediate Children for Restricted Return Values.....	182
To Search for All Children of an Entry in Sorted Order.....	182
To Limit the Number of Returned Search Entries and Search Time.....	182
To Get Information about How Indexes are used in a Search Operation.....	182
Working with the Matching Entry Count Control.....	184
Adding Entries.....	185
To Add an Entry Using an LDIF File.....	185
To Add an Entry Using the Changetype LDIF Directive.....	186
To Add Multiple Entries in a Single File.....	187
Deleting Entries Using ldapdelete.....	187
To Delete an Entry Using ldapdelete.....	188

- To Delete Multiple Entries Using an LDIF File..... 188
- Deleting Entries Using ldapmodify..... 188
 - To Delete an Entry Using ldapmodify..... 188
- Modifying Entries Using ldapmodify..... 188
 - To Modify an Attribute from the Command Line..... 189
 - To Modify Multiple Attributes in an Entry from the Command Line..... 189
 - To Add an Attribute from the Command Line..... 189
 - To Add an Attribute Using the Language Subtype..... 190
 - To Add an Attribute Using the Binary Subtype..... 190
 - To Delete an Attribute..... 190
 - To Delete One Value from an Attribute with Multiple Values..... 191
 - To Rename an Entry..... 191
 - To Move an Entry Within a Directory Server..... 191
 - To Move an Entry from One Machine to Another..... 192
 - To Move Multiple Entries from One Machine to Another..... 192
- Working with the Parallel-Update Tool..... 193
 - To Run the Parallel-Update Tool..... 193
- Working with the Watch-Entry Tool..... 194
 - To Run the Watch-Entry Tool..... 194
- Working with LDAP Transactions..... 195
 - To Request a Batched Transaction Using ldapmodify..... 195

Chapter 12: Working with Virtual Attributes..... 197

- Overview of Virtual Attributes..... 198
 - Viewing the List of Default Virtual Attributes..... 198
 - To View the List of Default Virtual Attributes Using dsconfig Non-Interactive Mode..... 199
- Viewing Virtual Attribute Properties..... 199
 - To View Virtual Attribute Properties..... 200
- Enabling a Virtual Attribute..... 200
 - To Enable a Virtual Attribute using dsconfig Interactive Mode..... 200
 - To Enable a Virtual Attribute Using dsconfig Non-Interactive Mode..... 201
- Creating User-Defined Virtual Attributes..... 201
 - To Create a User-Defined Virtual Attribute in Interactive Mode..... 201
 - To Create a User-Defined Virtual Attribute Using dsconfig in Non-Interactive Mode..... 203
- Creating Mirror Virtual Attributes..... 203
 - To Create a Mirror Virtual Attribute in Non-Interactive Mode..... 204
- Editing a Virtual Attribute..... 204
 - To Edit a Virtual Attribute Using dsconfig in Non-Interactive Mode..... 205
- Deleting a Virtual Attribute..... 205
 - To Delete a Virtual Attribute..... 205

Chapter 13: Working with Groups..... 207

- Overview of Groups..... 208
- About the isMemberOf and isDirectMemberOf Virtual Attribute..... 208
- Using Static Groups..... 210
 - Creating Static Groups..... 211
 - Searching Static Groups..... 212
- Using Dynamic Groups..... 214
 - Creating Dynamic Groups..... 215
 - Searching Dynamic Groups..... 216
 - Using Dynamic Groups for Internal Operations..... 218
- Using Virtual Static Groups..... 218
 - Creating Virtual Static Groups..... 219

Searching Virtual Static Groups.....	220
Creating Nested Groups.....	221
To Create Nested Static Groups.....	221
Maintaining Referential Integrity with Static Groups.....	222
Monitoring the Group Membership Cache.....	224
Using the Entry Cache to Improve the Performance of Large Static Groups.....	224
To Enable the Entry Cache.....	225
To Create Your Own Entry Cache for Large Groups.....	225
Monitoring the Entry Cache.....	225
Tuning the Index Entry Limit for Large Groups.....	226
Summary of Commands to Search for Group Membership.....	226
Migrating Sun/Oracle Groups.....	227
Migrating Static Groups.....	227
Migrating Static Groups to Virtual Static Groups.....	228
Migrating Dynamic Groups.....	229

Chapter 14: Encrypting Sensitive Data.....231

Encrypting and Protecting Sensitive Data.....	232
About the Encryption-Settings Database.....	232
Supported Encryption Ciphers and Transformations.....	232
Using the encryptions-settings Tool.....	233
Creating Encryption-Settings Definitions.....	234
Changing the Preferred Encryption-Settings Definition.....	234
Deleting an Encryption-Settings Definition.....	235
Configuring the Encryption-Settings Database.....	236
Backing Up and Restoring the Encryption-Settings Definitions.....	237
Exporting Encryption-Settings Definitions.....	237
Importing Encryption-Settings Definitions.....	238
Enabling Data Encryption in the Server.....	238
Using Data Encryption in a Replicated Environment.....	239
Dealing with a Compromised Encryption Key.....	239
Configuring Sensitive Attributes.....	240
To Create a Sensitive Attribute.....	242
Configuring Global Sensitive Attributes.....	243
To Configure a Global Sensitive Attribute.....	243
Excluding a Global Sensitive Attribute on a Client Connection Policy.....	243
To Exclude a Global Sensitive Attribute on a Client Connection Policy.....	243

Chapter 15: Working with the LDAP Changelog.....245

Overview of the LDAP Changelog.....	246
Key Changelog Features.....	246
Useful Changelog Features.....	248
Example of the Changelog Features.....	248
Viewing the LDAP Changelog Properties.....	250
To View the LDAP Changelog Properties Using dsconfig Non-Interactive Mode.....	250
Enabling the LDAP Changelog.....	250
To Enable the LDAP Changelog Using dsconfig Non-Interactive Mode.....	251
To Enable the LDAP Changelog Using Interactive Mode.....	251
Changing the LDAP Changelog Database Location.....	251
To Change the LDAP Changelog Location Using dsconfig Non-Interactive Mode.....	252
To Reset the LDAP Changelog Location Using dsconfig Non-Interactive Mode.....	252
Viewing the LDAP Changelog Parameters in the Root DSE.....	252
To View the LDAP Changelog Parameters.....	253

- Viewing the LDAP Changelog Using ldapsearch.....253
 - To View the LDAP Changelog Using ldapsearch..... 254
 - To View the LDAP Change Sequence Numbers..... 254
 - To View LDAP Changelog Monitoring Information.....255
- Indexing the LDAP Changelog..... 255
 - To Index a Changelog Attribute.....256
 - To Exclude Attributes from Indexing..... 256
- Tracking Virtual Attribute Changes in the LDAP Changelog.....256
 - To Track Virtual Attribute Changes in the LDAP Changelog..... 257

Chapter 16: Managing Access Control..... 259

- Overview of Access Control..... 260
 - Key Access Control Features..... 260
 - General Format of the Access Control Rules..... 262
 - Summary of Access Control Keywords..... 262
- Working with Targets..... 268
 - target..... 268
 - targetattr.....269
 - targetfilter..... 271
 - targetfilters..... 271
 - targetscope..... 272
 - targetcontrol.....273
 - extOp..... 273
- Examples of Common Access Control Rules..... 273
 - Administrator Access..... 273
 - Anonymous and Authenticated Access..... 274
 - Delegated Access to a Manager..... 274
 - Proxy Authorization..... 274
- Validating ACIs Before Migrating Data..... 275
 - To Validate ACIs from a File..... 275
 - To Validate ACIs in Another Directory Server..... 276
- Migrating ACIs from Sun/Oracle to Ping Identity Directory Server..... 277
 - Support for Macro ACIs..... 277
 - Support for the roleDN Bind Rule..... 277
 - Targeting Operational Attributes..... 277
 - Specification of Global ACIs..... 278
 - Defining ACIs for Non-User Content..... 278
 - Limiting Access to Controls and Extended Operations..... 278
 - Tolerance for Malformed ACI Values..... 278
 - About the Privilege Subsystem..... 279
 - Identifying Unsupported ACIs..... 279
- Working with Privileges..... 279
 - Available Privileges..... 280
 - Privileges Automatically Granted to Root Users..... 282
 - Assigning Additional Privileges for Administrators..... 283
 - Assigning Privileges to Normal Users and Individual Root Users..... 283
 - Disabling Privileges..... 284
- Working with Proxied Authorization..... 284
 - Configuring Proxied Authorization..... 285
 - Restricting Proxy Users..... 286
 - Restricting Proxied Authorization for Specific Users..... 288

Chapter 17: Managing the Schema..... 293

About the Schema.....	294
About the Schema Editor.....	294
Default Directory Server Schema Files.....	295
Extending the Directory Server Schema.....	295
General Tips on Extending the Schema.....	296
Managing Attribute Types.....	297
Attribute Type Definitions.....	298
Basic Properties of Attributes.....	298
Viewing Attributes.....	299
Creating a New Attribute over LDAP.....	301
To Add an New Attribute to the Schema over LDAP.....	301
To Add Constraints to Attribute Types.....	301
Managing Object Classes.....	302
Object Classes Types.....	302
Object Class Definition.....	303
Basic Object Class Properties.....	303
Viewing Object Classes.....	304
Managing an Object Class over LDAP.....	305
To Manage an Object Class over LDAP.....	305
Creating a New Object Class Using the Schema Editor.....	306
To Create a New Object Class Using the Schema Editor.....	306
Extending the Schema Using a Custom Schema File.....	307
To Extend the Schema Using a Custom Schema File.....	307
Managing Matching Rules.....	308
Matching Rule Definition.....	309
Default Matching Rules.....	309
Basic Matching Rule Properties.....	313
Viewing Matching Rules.....	313
Managing Attribute Syntaxes.....	314
Attribute Syntax Definition.....	314
Default Attribute Syntaxes.....	314
Basic Attribute Syntax Properties.....	317
Viewing Attribute Syntaxes.....	317
Using the Schema Editor Utilities.....	318
To Check Schema Compliance Using the Schema Editor.....	318
Modifying a Schema Definition.....	318
To Modify a Schema Definition.....	318
Deleting a Schema Definition.....	319
To Delete a Schema Definition.....	319
Schema Checking.....	319
To View the Schema Checking Properties.....	319
To Disable Schema Checking.....	319
Managing Matching Rule Uses.....	320
Matching Rule Use Definitions.....	320
To View Matching Rule Uses.....	321
Managing DIT Content Rules.....	321
DIT Content Rule Definitions.....	321
To View DIT Content Rules.....	322
Managing Name Forms.....	322
Name Form Definitions.....	322
To View Name Forms.....	322
Managing DIT Structure Rules.....	323
DIT Structure Rule Definition.....	323
To View DIT Structure Rules.....	323
Managing JSON Attribute Values.....	324
Configuring JSON Attribute Constraints.....	325

To Add Constraints to JSON Attributes..... 328

Chapter 18: Managing Password Policies..... 329

- Viewing Password Policies..... 330
 - To View Password Policies..... 330
 - To View a Specific Password Policy..... 331
- About the Password Policy Properties..... 331
- Modifying an Existing Password Policy..... 333
 - To Modify an Existing Password Policy..... 333
- Creating a New Password Policy..... 333
 - To Create a New Password Policy..... 333
 - To Assign a Password Policy to an Individual Account..... 334
 - To Assign a Password Policy Using a Virtual Attribute..... 334
- Deleting a Password Policy..... 335
 - To Delete a Password Policy..... 335
- Modifying a User’s Password..... 335
 - Validating a Password..... 336
 - Retiring a Password..... 336
 - To Change a User’s Password using the Modify Operation..... 337
 - To Change a User’s Password using the Password Modify Extended Operation..... 337
 - To Use an Automatically-Generated Password..... 337
- Enabling YubiKey Authentication..... 337
- Enabling Social Login..... 338
- Managing User Accounts..... 338
 - To Return the Password Policy State Information..... 338
 - To Determine Whether an Account is Disabled..... 339
 - To Disable an Account..... 339
 - To Enable a Disabled Account..... 339
 - To Assign the Manage-Account Access Privileges to Non-Root Users..... 339
- Disabling Password Policy Evaluation..... 341
 - To Globally Disable Password Policy Evaluation..... 341
 - To Exempt a User from Password Policy Evaluation..... 341
- Managing Password Validators..... 341
 - Password Validators..... 342
 - Configuring Password Validators..... 343

Chapter 19: Managing Replication..... 349

- Overview of Replication..... 350
- Replication Versus Synchronization..... 351
- Replication Terminology..... 351
- Replication Architecture..... 353
 - Eventual Consistency..... 354
 - Replicas and Replication Servers..... 354
 - Authentication and Authorization..... 355
 - Logging..... 355
- Replication Deployment Planning..... 355
 - Location..... 356
 - User-Defined LDAP..... 356
 - Disk Space..... 356
 - Memory..... 356
 - Time Synchronization..... 357
 - Communication Ports..... 357
 - Hardware Load Balancers..... 357

Directory Proxy Server.....	357
To Display the Server Information for a Replication Deployment.....	357
To Display All Status Information for a Replication Deployment.....	358
Enabling Replication.....	358
Overview.....	358
Command Line Interface.....	359
What Happens When You Enable Replication.....	359
Initialization.....	360
Deploying a Basic Replication Topology.....	360
To Deploy a Basic Replication Deployment.....	360
A Deployment with Non-Interactive dsreplication.....	363
To Deploy with Non-Interactive dsreplication.....	363
To Use dsreplication with SASL GSSAPI (Kerberos).....	364
Configuring Assured Replication.....	366
About the Replication Assurance Policy.....	366
Points about Assured Replication.....	368
To Configure Assured Replication.....	369
About the Assured Replication Controls.....	372
Managing the Topology.....	372
To Add a Server to the Topology.....	372
Disabling Replication and Removing a Server from the Topology.....	373
Replacing the Data for a Replicating Domain.....	373
To Replace the Data.....	374
Advanced Configuration.....	374
Changing the replicationChanges DB Location.....	374
To Change the replicationChanges DB Location.....	375
Modifying the Replication Purge Delay.....	375
To Modify the Replication Purge Delay.....	375
Configuring a Single Listener-Address for the Replication Server.....	376
To Configure a Replication Server to Listen on a Single Address.....	376
Monitoring Replication.....	376
Monitoring Replication Using cn=monitor.....	377
Replication Best Practices.....	378
About the dsreplication Command-Line Utility.....	378
Replication Conflicts.....	379
Types of Replication Conflicts.....	380
Naming Conflict Scenarios.....	380
Modification Conflict Scenarios.....	382
Troubleshooting Replication.....	383
Recovering a Replica with Missed Changes.....	383
Performing a Manual Initialization.....	384
Fixing Replication Conflicts.....	384
To Fix a Modify Conflict.....	385
To Fix a Naming Conflict.....	385
Fixing Mismatched Generation IDs.....	386
Replication Reference.....	386
Summary of the dsreplication Subcommands.....	386
Summary of the Direct LDAP Monitor Information.....	388
Summary of the Indirect LDAP Server Monitor Information.....	390
Summary of the Remote Replication Server Monitor Information.....	391
Summary of the Replica Monitor Information.....	394
Summary of the Replication Server Monitor Information.....	395
Summary of the Replication Server Database Monitor Information.....	396
Summary of the Replication Server Database Environment Monitor Information.....	396
Summary of the Replication Summary Monitor Information.....	400
Summary of the replicationChanges Backend Monitor Information.....	401

Summary of the Replication Protocol Buffer Monitor Information.....	401
Advanced Topics Reference.....	402
About the Replication Protocol.....	402
Change Number.....	404
Conflict Resolution.....	404
WAN-Friendly Replication.....	405
WAN Gateway Server.....	405
WAN Message Routing.....	406
WAN Gateway Server Selection.....	407
WAN Replication in Mixed-Version Environments.....	408
Recovering a Replication Changelog.....	408
Disaster Recovery.....	408
Chapter 20: Managing Logging.....	411
Default Directory Server Logs.....	412
Types of Log Publishers.....	412
Viewing the List of Log Publishers.....	414
Enabling or Disabling a Default Log Publisher.....	414
Managing Access and Error Log Publishers.....	415
Managing File-Based Access Log Publishers.....	415
Access Log Format.....	416
Access Log Example.....	417
Modifying the Access Log Using dsconfig Interactive Mode.....	417
Modifying the Access Log Using dsconfig Non-Interactive Mode.....	418
Modifying the Maximum Length of Log Message Strings.....	418
Generating Access Logs Summaries.....	418
To Generate an Access Log Summary.....	419
About Log Compression.....	420
About Log Signing.....	421
To Configure Log Signing.....	421
To Validate a Signed File.....	421
Creating New Log Publishers.....	422
To Create a New Log Publisher.....	422
To Create a Log Publisher Using dsconfig Interactive Command-Line Mode.....	423
Configuring Log Rotation.....	423
To Configure the Log Rotation Policy.....	424
Configuring Log Rotation Listeners.....	424
Configuring Log Retention.....	425
To Configure the Log Retention Policy.....	425
Configuring Filtered Logging.....	425
To Configure a Filtered Log Publisher.....	426
Managing Admin Alert Access Logs.....	426
About Access Log Criteria.....	426
Configuring an Admin Alert Access Log Publisher.....	427
Managing Syslog-Based Access Log Publishers.....	428
Before You Begin.....	429
Default Access Log Severity Level.....	429
Syslog Facility Properties.....	429
Queue-Size Property.....	430
Configuring a Syslog-Based Access Log Publisher.....	430
Managing the File-Based Audit Log Publishers.....	430
Audit Log Format.....	431
Audit Log Example.....	431
Enabling the File-Based Audit Log Publisher.....	432
Obscuring Values in the Audit Log.....	432

Managing the JDBC Access Log Publishers.....	432
Before You Begin.....	432
Configuring the JDBC Drivers.....	433
Configuring the Log Field Mapping Tables.....	433
Configuring the JDBC Access Log Publisher using dsconfig Interactive Mode.....	434
Configuring the JDBC Access Log Publisher Using dsconfig Non-Interactive Mode.....	436
Managing the File-Based Error Log Publisher.....	436
Error Log Example.....	437
To Modify the File-Based Error Logs.....	437
Managing the Syslog-Based Error Log Publisher.....	437
Syslog Error Mapping.....	438
Configuring a Syslog-Based Error Log Publisher.....	438
Creating File-Based Debug Log Publishers.....	438
To Create a File-Based Debug Log Publisher.....	439
To Delete a File-Based Debug Log Publisher.....	439

Chapter 21: Managing Monitoring..... 441

The Monitor Backend.....	442
Monitoring Disk Space Usage.....	443
Monitoring with the Data Metrics Server.....	445
About the Collection of System Monitoring Data.....	445
Monitoring Key Performance Indicators by Application.....	446
Configuring the External Servers.....	446
Preparing the Servers Monitored by the Data Metrics Server.....	446
Configuring the Processing Time Histogram Plugin.....	447
Setting the Connection Criteria to Collect SLA Statistics by Application.....	448
Updating the Global Configuration.....	448
Proxy Considerations for Tracked Applications.....	449
Monitoring Using SNMP.....	449
SNMP Implementation.....	449
Configuring SNMP.....	450
To Configure SNMP.....	450
Configuring SNMP on AIX.....	452
SNMP on AIX Security Considerations.....	453
MIBS.....	453
Monitoring with the Administrative Console.....	454
To View the Monitor Dashboard.....	454
Accessing the Processing Time Histogram.....	454
To Access the Processing Time Histogram.....	454
Monitoring with JMX.....	454
Running JConsole.....	455
To Run JConsole.....	455
Monitoring the Directory Server Using JConsole.....	456
To Monitor the Directory Server using JConsole.....	456
Monitoring Using the LDAP SDK.....	458
Monitoring over LDAP.....	458
Profiling Server Performance Using the Stats Logger.....	459
To Enable the Stats Logger.....	459
To Configure Multiple Periodic Stats Loggers.....	460
Adding Custom Logged Statistics to a Periodic Stats Logger.....	461
To Configure a Custom Logged Statistic Using dsconfig Interactive.....	462
To Configure a Custom Stats Logger Using dsconfig Non-Interactive.....	463

Chapter 22: Managing Notifications and Alerts..... 465

- Working with Account Status Notifications..... 466
 - Account Status Notification Types.....466
 - Working with the Error Log Account Status Notification Handler..... 467
 - Working with the SMTP Account Status Notification Handler.....467
 - Associating Account Status Notification Handlers with Password Policies..... 469
- Working with Administrative Alert Handlers..... 469
 - Administrative Alert Types..... 470
- Configuring the JMX Connection Handler and Alert Handler..... 470
 - To Configure the JMX Connection Handler.....470
 - To Configure the JMX Alert Handler.....471
- Configuring the SMTP Alert Handler.....471
 - Configuring the SMTP Alert Handler.....471
- Configuring the SNMP Subagent Alert Handler..... 471
 - To Configure the SNMP Subagent Alert Handler..... 471
- Working with the Alerts Backend.....472
 - To View Information in the Alerts Backend..... 472
 - To Modify the Alert Retention Time.....472
 - To Configure Duplicate Alert Suppression.....473
- Working with Alarms, Alerts, and Gauges.....473
 - To View Information in the Alarms Backend.....474
- Testing Alerts and Alarms.....474
 - To Test Alarms and Alerts.....474
 - Indeterminate Alarms.....476

Chapter 23: Managing the SCIM Servlet Extension.....477

- Overview of SCIM Fundamentals.....478
 - Summary of SCIM Protocol Support.....478
 - About the Identity Access API.....479
- Configuring SCIM.....479
 - Creating Your Own SCIM Application.....479
 - Configuring the SCIM Servlet Extension.....480
 - SCIM Servlet Extension Authentication.....483
 - Verifying the SCIM Servlet Extension Configuration.....484
- Configuring Advanced SCIM Extension Features.....485
 - Managing the SCIM Schema.....485
 - Mapping SCIM Resource IDs.....491
 - Using Pre-defined Transformations.....492
 - Mapping LDAP Entries to SCIM Using the SCIM-LDAP API.....492
 - SCIM Authentication.....493
 - SCIM Logging.....493
 - SCIM Monitoring.....493
- Configuring the Identity Access API.....494
 - To Configure the Identity Access API.....494
 - To Disable Core SCIM Resources.....495
 - To Verify the Identity Access API Configuration.....495
- Monitoring the SCIM Servlet Extension.....495
 - Testing SCIM Query Performance.....495
 - Monitoring Resources Using the SCIM Extension.....496
 - About the HTTP Log Publishers.....498

Chapter 24: Managing Server SDK Extensions.....	501
About the Server SDK.....	502
Available Types of Extensions.....	502
Chapter 25: Troubleshooting the Server.....	505
Working with the Collect Support Data Tool.....	506
Server Commands Used in the Collect Support Data Tool.....	506
JDK Commands Used in the Collect-Support-Data Tool.....	506
Linux Commands Used in the collect-support-data Tool.....	507
Solaris Commands Used in the collect-support-data Tool.....	507
AIX Commands Used in the collect-support-data Tool.....	508
MacOS Commands Used in the Collect Support Data Tool.....	508
Available Tool Options.....	509
To Run the Collect Support Data Tool.....	510
Directory Server Troubleshooting Information.....	510
Error Log.....	510
server.out Log.....	511
Debug Log.....	512
Replication Repair Log.....	513
Config Audit Log and the Configuration Archive.....	513
Access and Audit Log.....	513
Setup Log.....	515
Tool Log.....	515
je.info and je.config Files.....	515
LDAP SDK Debug Log.....	516
About the Monitor Entries.....	516
Directory Server Troubleshooting Tools.....	517
Server Version Information.....	517
LDIF Connection Handler.....	517
dbtest Tool.....	518
Index Key Entry Limit.....	518
Embedded Profiler.....	519
Oracle Berkeley DB Java Edition Utilities.....	520
Troubleshooting Resources for Java Applications.....	520
Java Troubleshooting Tools.....	520
Java Diagnostic Information.....	523
Java Troubleshooting Tools (IBM JDK).....	523
Troubleshooting Resources in the Operating System.....	524
Identifying Problems with the Underlying System.....	524
Examining CPU Utilization.....	524
Examining Disk Utilization.....	525
Examining Process Details.....	526
Tracing Process Execution.....	527
Problems with SSL Communication.....	528
Examining Network Communication.....	528
Common Problems and Potential Solutions.....	529
General Methodology to Troubleshoot a Problem.....	529
The Server Will Not Run Setup.....	530
The Server Will Not Start.....	531
The Server Has Crashed or Shut Itself Down.....	535
Conditions for Automatic Server Shutdown.....	536
The Server Will Not Accept Client Connections.....	536

The Server is Unresponsive.....	537
The Server is Slow to Respond to Client Requests.....	538
The Server Returns Error Responses to Client Requests.....	539
The Server Must Disconnect a Client Connection.....	540
The Server is experiencing problems with replication.....	541
How to Regenerate the Server ads-certificate.....	541
The Server behaves differently from Sun/Oracle.....	542
Troubleshooting ACI Evaluation.....	542
Problems with the Administrative Console.....	543
Problems with the Administrative Console: JVM Memory Issues.....	544
Problems with the HTTP Connection Handler.....	544
Virtual Process Size on RHEL6 Linux is Much Larger than the Heap.....	545
AIX Command-Line Tools Not Working with DNS using Special Characters.....	545
Providing Information for Support Cases.....	545
Chapter 26: Command-Line Tools.....	547
Using the Help Option.....	548
Available Command-Line Utilities.....	548
Managing the tools.properties File.....	551
Creating a Tools Properties File.....	551
Tool-Specific Properties.....	552
Specifying Default Properties Files.....	552
Evaluation Order Summary.....	553
Evaluation Order Example.....	553
Running Task-based Utilities.....	554

Preface

This guide presents the procedures and reference material necessary to install, administer and troubleshoot the Ping Identity Directory Server in multi-client, high-load production environments.

Purpose of This Guide

The purpose of this guide is to provide procedures and concepts that can be used to manage the Ping Identity® Directory Server in a multi-client environment. It also provides information to monitor and set up the necessary logs needed to troubleshoot the server's performance.

The Directory Server is part of the PingData Platform. The PingData Platform is the consumer-grade identity access and management platform—built specifically to handle the massive scale and real-time demands of hundreds of millions of customers. It delivers a consistent, seamless, personalized brand experience that makes each customer feel valued.

The PingData Platform provides a unified view of customer data across all applications, channels, partners, and lines of business. The result is:

- Increased customer trust and confidence through greater transparency and customer control of personal data.
- A consistent, personalized customer experience that promotes better conversion, up-selling, and cross-selling.

Audience

The guide is intended for administrators responsible for installing, maintaining, and monitoring servers in large-scale, high load production environments. It is assumed that the reader has the following background knowledge:

- > PingData Platforms and LDAPv3 concepts
- > System administration principles and practices
- > Understanding of Java VM optimization and garbage collection processes
- > Application performance monitoring tools

Related Documentation

The following list shows the full documentation set that may help you manage your deployment:

- > *Ping Identity® Directory Server Administration Guide*
- > *Ping Identity® Directory Server Reference Guide (HTML)*
- > *Ping Identity® Directory Proxy Server Administration Guide*

- > *Ping Identity® Directory Proxy Server Reference Guide (HTML)*
- > *PingData® Data Sync Server Administration Guide*
- > *PingData® Data Sync Server Reference Guide (HTML)*
- > *PingData® Data Metrics Server Administration Guide*
- > *PingData® Data Governance Broker Administration Guide*
- > *PingData Security Guide*
- > *UnboundID® LDAP SDK*
- > *UnboundID® Server SDK*

Document Conventions

The following table shows the document convention used in this guide.

Convention	Usage
Monospace	Commands, filenames, directories, and file paths
Monospace Bold	User interface elements, menu items and buttons
<i>Italic</i>	Identifies file names, doc titles, terms, variable names, and emphasized text

Chapter

1 Overview of the Server

The Ping Identity® Directory Server is a high performance, extensible directory and PingData Platform, written completely in Java™. The Directory Server centralizes consumer and user identity management information, subscriber data management, application configurations, and user credentials into a network, enterprise, or virtualized database environment. It provides seamless data management over a distributed system based on a standardized solution that meets the constant performance demands for today's markets. The Directory Server simplifies administration, reduces costs, and secures information in systems that scale for very large numbers of users.

This chapter provides an overview of the Directory Server's features and components.

Topics:

- [Server Features](#)
- [Administration Framework](#)
- [Server Tools Location](#)

Server Features

The Ping Identity Directory Server is a powerful, 100% Java, production-proven PingData Platform solution for mission-critical and large-scale applications. The Directory Server provides an extensive feature-rich set of tools that can meet the production needs of your system.

- **Full LDAP Version 3 Implementation.** The Directory Server fully supports the Lightweight Directory Access Protocol version 3 (LDAP v3), which supports the Request For Comments (RFCs) specified in the protocol. The Directory Server provides a feature-rich solution that supports the core LDAPv3 protocol in addition to server-specific controls and extended operations.
- **High Availability.** The Directory Server supports N-way multi-master replication that eliminates single points of failure and ensures high availability for a networked topology. The Directory Server allows data to be stored across multiple machines and disk partitions for fast replication. The Directory Server also supports replication in entry-balancing proxy server deployments.
- **Administration Tools.** The Directory Server provides a full set of command-line tools, an Administrative Console, and a Java-based setup tool to configure, monitor, and manage any part of the server. The Directory Server has a task-based subsystem that provides automated scheduling of basic functions, such as backups, restores, imports, exports, restarts, and shutdowns. The set of utilities also includes a troubleshooting support tool that aggregates system metrics into a zip file, which administrators can send to your authorized support provider for analysis.
- **Security Mechanisms.** The Directory Server provides extensive security mechanisms to protect data and prevent unauthorized access. Access control list (ACL) instructions are available down to the attribute value level and can be stored within each entry. The Directory Server allows connections over Secure Sockets Layer (SSL) through an encrypted communication tunnel. Clients can also use the StartTLS extended operation over standard, non-encrypted ports. Other security features include a privilege subsystem for fine-grained granting of rights, a password policy subsystem that allows configurable password validators and storage schemes, and SASL authentication mechanisms to secure data integrity, such as PLAIN, ANONYMOUS, EXTERNAL, CRAM-MD5, Digest-MD5, and GSSAPI. The Directory Server also supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes. See the *Ping Identity Security Guide* for details.
- **Monitoring and Notifications.** The Directory Server supports monitoring entries using the Data Metrics Server, JConsole, Simple Network Management Protocol (SNMP), or using the Administrative Console. Administrators can track the response times for LDAP operations using a monitoring histogram as well as record performance statistics down to sub-second granularity. The Directory Server also supports configurable notifications, auditing, and logging subsystems with filtered logging capabilities.
- **Powerful LDAP SDK.** The Directory Server is based on a feature-rich LDAP SDK for Java, designed by PingData. The UnboundID® LDAP SDK is a Java API standard that overcomes the many limitations of the Java Naming and Directory Interface (JNDI) model. For example, JNDI does not address the use of LDAP controls and extended operations. The LDAP SDK for Java provides support for controls and extended operations to leverage the Directory Server's extensible architecture for their applications.

The Standard LDAP SDK for Java is available for free to public users and can be downloaded from the UnboundID web site. The Commercial version of the LDAP SDK for Java is part of the Directory Server distribution. For more information on the LDAP SDK for Java, to go <http://www.unboundid.com/products/ldapsdk>.

- **SCIM Extension.** The Directory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure and fast manner.
- **Server SDK.** PingData also provides the Server SDK, which is a library of Java packages, classes, and build tools to help in-house or third-party developers create client extensions for the Ping Identity® Directory Server, Ping Identity® Directory Proxy Server, and PingData® Data Sync Server. The servers were designed with a highly extensible and scalable architecture with multiple plug-in points for your customization needs. The Server SDK provides APIs to alter the behavior of each server's components without affecting its code base.
- **Multi-Platform Support.** The Ping Identity Directory Server is a pure Java application and is certified VMWare Ready™. It is intended to run within the Java Virtual Machine on any Java Standard Edition (SE) or Enterprise Edition (EE) certified platform. For the list of supported platforms and Java versions, access your Customer Support Center portal or contact your authorized support provider.

Any known OS or JDK-related issues will be documented in the release notes distributed with the product. Direct any questions or requests for additional platform certifications to your authorized support provider.

Administration Framework

The Directory Server provides an administration and configuration framework capable of managing stand-alone servers, server groups, and highly-available deployments that include multiple redundant server instances. Administrators can configure changes locally or remotely on a single server or on all servers in a server group. Each server configuration is stored as a flat file (LDIF) that can be accessed under the `cn=config` branch of the Directory Information Tree (DIT). Administrators can tune the configuration and perform maintenance functions over LDAP using a suite of command-line tools, a GUI-based Java console (for installs, uninstalls, and server status), or an Administrative Console (for configuration and monitoring). The Directory Server also provides plug-ins to extend the functionality of its components.

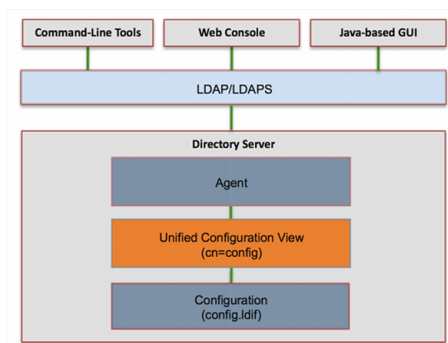


Figure 1: Directory Server Configuration Network

Server Tools Location

PingData distributes the Directory Server, Administrative Console, and LDAP SDK for Java in zip format. After unzipping the file, you can access the `setup` utility in the server root directory, located at `PingDirectory`. The Directory Server stores a full set of command-line tools for maintaining your system in the `PingDirectory/bin` directory for UNIX[®] or Linux[®] machines and the `PingDirectory\bat` directory for Microsoft[®] Windows[®] machines.

Prior to installing the directory server, read Chapter 2 *Preparing Your Environment*, which presents important information on setting up your machines. Chapter 3 *Installing the Directory Server* presents procedures to install a server instance using the `setup` utility. This utility can be run in one of the two available installation modes: interactive command-line, and non-interactive command-line. Chapter 4 *Configuring the Directory Server* provides procedures to modify the configuration of a server instance or a group of servers using the command-line tools and the Administrative Console.

Chapter

2

Preparing Your Environment

The Ping Identity Directory Server offers a highly portable and scalable architecture that runs on multiple platforms and operating systems. The Directory Server is specifically optimized for those operating systems used in environments that process a very large number of entries.

This chapter presents some procedures to set up your server machines for optimal processing efficiency.

Topics:

- [*Before You Begin*](#)
- [*Preparing the Operating System \(Solaris\)*](#)
- [*Preparing the Operating System \(Linux\)*](#)
- [*Running as a Non-Root User*](#)

Before You Begin

The Directory Server requires certain software packages for the proper operation of the server. For optimized performance, the Ping Identity Directory Server requires Java for 64-bit architectures. To view the minimum required Java version, access your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

It is also highly recommended that a Network Time Protocol (NTP) system be in place so that multi-server environments are synchronized and timestamps are accurate.

Installing Java

For optimized performance, the Ping Identity Directory Server requires Java for 64-bit architectures. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the Ping Identity Directory Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Server. This setup requires that the JDK, rather than the JRE, for the 64-bit version, be downloaded.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the Oracle download site.
2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.
3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

To Install Java (IBM)

1. Open a browser and navigate to the IBM download site.
2. Select the Java version for your operating system.

Preparing the Operating System (Solaris)

The Ping Identity Directory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance. These optimizations include using the ZFS filesystem on Solaris systems, restricting ZFS

memory consumption, limiting transaction group writes, using compression and disabling access time updates.

Using ZFS

PingData strongly recommends the use of ZFS™ as the underlying filesystem on Solaris 10 and OpenSolaris systems. ZFS is a 128-bit filesystem that can store billions of times more data than traditional 64-bit systems. Based on a storage pool model, ZFS aggregates devices (mirrors, RAID-Z with single or double parity, concatenated or striped storage) into a virtual data source from which filesystems can be constructed. ZFS provides excellent performance, end-to-end data integrity, simple administration management, and unmatched scalability. It also provides many useful features, such as automatic checksum, dynamic striping, variable block sizes, compression, and unlimited constant-time snapshots. ZFS is part of the Solaris 10 and OpenSolaris operating systems.

All of the Directory Server's components should be located on a single storage pool (zpool), rather than having separate pools configured for different server components (for example, one pool for the database and a second for log files). Single zpool configurations are the simplest and easiest to manage. From there, you can create multiple filesystems inside the pool and optionally reserve space for one or more of the filesystems.

ZFS's copy-on-write transactional model does not require isolating I/O-intensive components. Therefore, all available disks should be placed in the same zpool, so that as many underlying spindles as possible can be used to provide the configuration with the greatest number of I/O operations per second.

To Restrict ZFS Memory Consumption

Despite its excellent performance, ZFS does not release memory fast enough for some LDAP operations that might need it. This delay could cause some processes to fail to start while attempting to allocate a large amount of memory for a JVM heap.

To curb memory allocation problems, make sure that the system is configured to limit the amount of memory for caching (for example, up to two gigabytes). The Directory Server relies on database caching rather than filesystem caching for its performance. Thus, the underlying system should be configured, so that the memory used by ZFS will not interfere with the memory used by the Directory Server. In most environments, we recommend that systems be configured to allow ZFS to use no more than 2 GB of memory for caching.

1. Open the `/etc/system` file.
2. ZFS caches data from all active storage pools in the ARC cache. We can limit its memory consumption by setting the maximum size of the ARC caches using the `zfs_arc_max` property. For example, add the following line to the end of the `/etc/system` file.

```
set zfs:zfs_arc_max= 0x80000000
```

This property sets the maximum size of the ARC cache to 2 GB (0x80000000 or 2147483648 bytes) for ZFS. Note that your system may require a different value.

3. If your system processes large write operations, see the section on Limiting ZFS Transaction Group Writes. Otherwise, reboot the machine for the change to take effect. Also note that this operation requires Solaris 10 update 4 (08/07) and Nevada (build 51) release or later.

To Limit ZFS Transaction Group Writes

PingData has found that the Directory Server can exhibit uneven throughput performance during continuous write loads for Oracle Berkeley DB Java Edition backends on ZFS systems. We have found that the ZFS Write Throttle feature stalls write operations when transaction groups are flushed to disk. During these periods, operation throughput can drop significantly with these large I/O bursts.

To smooth out write throughput and improve latency, we recommend setting the `zfs_write_limit_override` property in the `etc/system` file to the size of the available disk cache on the system.

1. Open the `/etc/system` file.
2. Add the following line to the end of the file. Set the value to the size of your onboard cache. For example, for a system that has a 32MB cache per disk, set the following parameter:

```
set zfs:zfs_write_limit_override=0x2000000
```

3. For the change to take effect, reboot the machine. Also note that this operation requires Solaris 10 update 4 (08/08) or later.

ZFS Access to Underlying Disks

Storage requirements vary depending on whether ZFS has access to the underlying disks. If possible, ZFS should be given direct access to the underlying disks that will be used to back the storage. Direct access to the underlying disks makes it possible to configure the system with the greatest degree of reliability and flexibility.

To configure the system, ZFS should be given direct access to the underlying disks that will be used to back the storage. In this configuration, the `zpool` used for the Directory Server should have a RAID 1+0 configuration (a stripe across one or more 2-disk mirrors). Although this setup reduces the amount of available space when compared with other configurations, like RAID-Z (ZFS data-parity scheme with full dynamic stripe width) or RAID-Z2 (ZFS dual parity RAID-Z), RAID 1+0 provides dramatically better performance and reliability.

If ZFS cannot get direct access to the underlying disks (for example, the system only has access to a logical unit number, LUN, on a storage area network, SAN), then the provided storage should already include some level of redundancy. Again, the RAID 1+0 configuration is recommended over other schemes like, RAID 5 or RAID 6. If the storage includes redundancy, then the `zpool` should be created with only that LUN and should not add any additional redundancy. In such a configuration, ZFS is not able to take advantage of its advanced self-healing capabilities when it detects any corruption at the filesystem level. However, ZFS check-summing can still detect those types of problems.

Configuring ZFS Compression

The ZFS filesystem should have compression enabled to improve performance as it reduces the amount of data that needs to be written or read from the underlying disks. In most cases, the reduced costs of the disk I/O outweighs the CPU cost of compressing and decompressing the data.

The following procedure assumes that the ZFS filesystem is named `ds`. The changes take effect immediately with no need to reboot or perform any other action.

Caution:



Knowing the actual size of files is useful when you need to back up files to a non-ZFS filesystem or estimate the amount of memory dedicated to caching. On traditional UNIX filesystems, the `du` command reports the sum of all the specified file sizes. However, on ZFS, `du` reports the amount of disk space consumed, which might not equal the sum of the file sizes if features like compression or multiple copies are enabled. Administrators should be aware of this difference when determining the database size using `du`.

Instead of using `du`, Ping Identity Directory Server provides a utility, `bin/sum-file-sizes`, that determines the size (in bytes, kilobytes, megabytes, or gigabytes) of the sum of a set of files even if ZFS compression or multiple copies are enabled.

To Configure ZFS Compression

- Turn on ZFS compression by running the `zfs` command.

```
# zfs set compression=on ds
```

To Disable the Access Time Update for Reads

You must have ZFS installed and configured on your system. Assume that the ZFS filesystem is named `ds`.

You should disable the access time update (*atime*) tracking, so that the kernel will not update the access time of a file every time it is requested. ZFS stores the filesystem attributes within the filesystem itself, so that you can access the attributes.

1. Disable the access time update for read operations.

```
# zfs set atime=off ds
```

2. To view all filesystem attributes, use the following command:

```
# zfs get all ds
```

Preparing the Operating System (Linux)

The Ping Identity Directory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance. These optimizations include increasing the file descriptor limit on Linux systems, setting filesystem flushes, editing OS-level environment variables, downloading some useful monitoring tools for Redhat Linux systems, and configuring for Huge Page support.

Configuring the File Descriptor Limits

The Ping Identity Directory Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. If needed, increase the file descriptor limit on the operating system.

If the operating system relies on `systemd`, refer to the Linux operating system documentation for instructions on setting the file descriptor limit.

To Set the File Descriptor Limit (Linux)

The Directory Server allows for an unlimited number of connections by default but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the server if it needs to handle a large number of concurrent connections.

Once the operating system limit is set, the number of file descriptors that the server will use can be configured by either using a `NUM_FILE_DESCRIPTOR` environment variable, or by creating a `config/num-file-descriptors` file with a single line such as, `NUM_FILE_DESCRIPTOR=12345`. If these are not set, the default of 65535 is used. This is strictly optional if wanting to ensure that the server shuts down safely prior to reaching the file descriptor limit.

1. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

```
ulimit -aH
```

2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to at least 65535. If there is no line that sets a value for this property, add the following to the end of the file:

```
fs.file-max = 65535
```

3. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before “#End of file”). Also note that you should insert a tab, rather than spaces, between the columns.

```
* soft nofile 65535
* hard nofile 65535
```


4. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

```
# ulimit -n
```

Note: For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits:



```
/etc/security/limits.d/20-nproc.conf
```

Add or edit the following lines if they do not already exist:

```
*      soft  nproc   65536
*      soft  nofile  65536
*      hard  nproc   65536
*      hard  nofile  65536
root   soft  nproc   unlimited
```

File System Tuning

Newer ext4 systems use delayed allocation to improve performance. This delays block allocation until it writes data to disk. Delayed allocation improves performance and reduces fragmentation by using the actual file size to improve block allocation. This feature may cause a risk of data loss in cases where a system loses power before all of the data has been written to disk. This may occur if a program is replacing the contents of a file without forcing a write to the disk with `fsync`. Make sure the default `auto_da_alloc` option is enabled on ext4 filesystems.

Administrators can tune ext3 and ext4 filesystems by setting the filesystem flushes and `noatime` to improve server performance. The following changes can be made in the `/etc/fstab` file.

To Set the Filesystem Flushes

With the out-of-the-box settings on Linux systems running the ext3 filesystem, the data is only flushed to disk every five seconds. If the Directory Server is running on a Linux system using the ext3 filesystem, consider editing the mount options for that filesystem to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one second.

You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

To Set `noatime` on ext3 and ext 4 Systems

If you are using an ext3 or ext4 filesystem, it is recommended that you set `noatime`, which turns off any `atime` updates during read accesses to improve performance. You should also set the flush frequency to the `/etc/fstab` file. Doing the change via the mount command alone will not survive across reboots.

- Run the following command on an ext3 system.

```
# mount -t ext3 -o noatime /dev/fs1
```

- Run the following command on an ext34 system.

```
# mount -t ext4 -o noatime /dev/fs1
```

Setting the Maximum User Processes

On some Linux distributions (Redhat Enterprise Linux Server/CentOS 6.0 or later), the default maximum number of user processes is set to 1024, which is considerably lower than the same parameter on older distributions (e.g., RHEL/CentOS 5.x). The default value of 1024 leads to some JVM memory errors when running multiple servers on a machine due to each Linux thread being counted as a user process. (Note that this is not an issue on Solaris and AIX platforms as individual threads are not counted as user processes.)

At startup, the Directory Server and its tools automatically attempt to raise the maximum user processes limit to 16,383 if the value reported by `ulimit` is less than that. If, for any reason, the server is unable to automatically set the maximum processes limit to 16,383, an error message will be displayed. It is recommended that the limit be set explicitly in `/etc/security/limit.conf`. For example:

```
* soft nproc 65535
* hard nproc 65535
```

The (*) can be replaced with the name of the user under which the software will run. These settings can also be manually configured by setting the `NUM_USER_PROCESSES` environment variable to 16383 or by setting the same variable in a file named `config/num-user-processes`.

About Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Server explicitly overrides the values of key environment variables like `PATH`, `LD_LIBRARY_PATH`, and `LD_PRELOAD` to ensure that something set in the environments that are used to start the server does not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the `set_environment_vars` function of the `lib/_script-util.sh` script. You will need to stop (`bin/stop-ds`) and re-start (`bin/start-ds`) the server for the change to take effect.

Install `sysstat` and `pstack` (Red Hat)

For Red Hat® Linux systems, you should install a couple of packages, `sysstat` and `pstack`, that are disabled by default, but are useful for troubleshooting purposes in the event that a problem occurs. The troubleshooting tool `collect-support-data` uses the `iostat`, `mpstat`, and `pstack`

utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

```
$ sudo yum install sysstat gdb dstat -y
```

Install dstat (SUSE Linux)

The `dstat` utility is used by the `collect-support-data` tool and can be obtained from the OpenSUSE project website. The following example shows how to install the `dstat` utility on SuSE Enterprise Linux 11 SP2:

1. Login as Root.
2. Add the appropriate repository using the `zypper` tool.
3. Install the `dstat` utility.

```
$ zypper install dstat
```

Disable Filesystem Swapping

It is recommended that any performance tuning services like `tuned` be disabled. If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file. This file is located at `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

Omit vm.overcommit_memory

Administrators should be aware that an improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file can cause the `setup` or `start-ds` tool to fail.

For Linux systems, the `vm.overcommit_memory` property sets the kernel policy for memory allocations. The default value of 0 indicates that the kernel determines the amount of free memory to grant a `malloc` call from an application. If the property is set to a value other than zero, it could lead the operating system to grab too much memory, depriving memory for the `setup` or `start-ds` tool.

We recommend omitting the property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing System Entropy

Entropy is used to calculate random data that is used by the system in cryptographic operations. Some environments with low entropy may have intermittent performance issues with SSL-based communication. This is more typical on virtual machines, but can occur in physical instances as well. Monitor the `kernel.random.entropy_avail` in `sysctl` value for best results.

If necessary, update `$JAVA_HOME/jre/lib/security/java.security` to use `file:/dev/./urandom` for the `securerandom.source` property.

Running as a Non-Root User

The drawback to not running as root is the inability to use network port numbers below 1024. Some operating systems provide workarounds for this limitation, but the best practice is to install and run the server as a user, other than root, and select port numbers greater than 1024.

On systems running Solaris 10 and OpenSolaris, you can use the User and Process Rights Management subsystem with the Role-Based Access Control (RBAC) mechanisms to grant users or roles only the privileges necessary to accomplish a specific task. Using RBAC avoids the assignment of full super-user (root) privileges to the user. For example, you can grant the `net_privaddr` privilege to a non-root user, or role, that gives him or her the ability to listen on privileged ports (for example, on ports 1024 or below). Similarly, granting the `sys_resource` privilege allows a user to bypass restrictions on resource limits, such as the number of file descriptors a process might use.

The Solaris User and Process Rights Management system can also be used to remove capabilities from users. For example, removing the `proc_info` privilege from a user prevents the user from seeing processes owned by other users. Removing the `file_link_any` privilege can prevent users from creating hard links to files owned by other users. Hard links are not needed by the Directory Server and can represent a security risk under certain conditions. The following table summarizes the Solaris privileges that you may want to assign to non-root users.

Privilege	Description
<code>net_privaddr</code>	Provides the ability to listen on privileged network ports.
<code>sys_resource</code>	Provides the ability to bypass restrictions on resource limits (including the number of available file descriptors).
<code>proc_info</code>	Provides the ability for users to see processes owned by other users on the system. This privilege is available to all users by default, but it can pose a security risk in some cases. PingData recommends that it be removed from the role used by the Directory Server.
<code>file_link_any</code>	Provides the ability to create hard links to files owned by other users on the system. This privilege is available to all users by default, but it can pose a security risk in some cases. PingData recommends that it be removed from the role used by the Directory Server.

Running as a Non-Root User (Linux)

To run as a non-root user but still allow connections on a privileged port, two options are available:

- **Use a Load-Balancer or Directory Proxy Server.** In many environments, the server can be run on a non-privileged port but can be hidden by a hardware load-balancer or LDAP Directory Proxy Server.
- **Use `netfilter`.** The `netfilter` mechanism, exposed through the `iptables` command, can be used to automatically redirect any requests from a privileged port to the unprivileged port on which the server is listening.

Enabling the Server to Listen on Privileged Ports (Linux)

Linux systems do not provide a direct analog to the Solaris User and Process Rights Management subsystems. Linux does have a similar mechanism called capabilities used to grant specific commands the ability to do things that are normally only allowed for a root account. This is different from the Solaris model because instead of granting the ability to a specific user, capabilities are granted to a specific command. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

```
$ sudo setcap cap_net_bind_service=+eip /ds/java/bin/java
```

The `java` binary needs an additional shared library (`libjli.so`) as part of the Java installation. More strict limitations are imposed on where the operating system will look for shared libraries to load for commands that have capabilities assigned. So it is also necessary to tell the operating system where to look for this library. This can be done by creating the file `/etc/ld.so.conf.d/libjli.conf` with the path to the directory that contains the `libjli.so` file. For example, if the Java installation is in `/ds/java`, the contents of that file should be:

```
/ds/java/lib/amd64/jli
```

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Creating a Solaris Role

To give multiple administrators access to the Directory Server, Ping Identity Directory Server recommends that a Solaris role be created to run the server and that all necessary administrators be added to that role. The Solaris role provides an audit trail that can be used to identify which administrator performed a given action, while still allowing administrators to run the server, to view and edit files used by the server, and to execute commands as that same user. As with normal user accounts, roles can be assigned privileges. The role used for the Directory Server should include the `net_privaddr` and `sys_resource` privileges and should exclude the `proc_info` and `file_link_any` privileges for improved security (that is, to eliminate the need for root access).

To Create a Solaris Role for Multiple Administrators

To give multiple administrators access to the Directory Server, Ping Identity Directory Server recommends that a Solaris role be created to run the server and that all necessary administrators be added to that role. The Solaris role provides an audit trail that can be used to identify which administrator performed a given action, while still allowing administrators to run the server, to view and edit files used by the server, and to execute commands as that same user. As with normal user accounts, roles can be assigned privileges. The role used for the Directory

Server should include the `net_privaddr` and `sys_resource` privileges and should exclude the `proc_info` and `file_link_any` privileges for improved security (that is, to eliminate the need for root access).

1. Create a Solaris role. Assume the role is named `ds` with all of the appropriate privileges needed to run the Directory Server. Make sure to enter the whole command on a single line.

```
# roleadd -d /export/home/ds -m -s /usr/bin/bash \  
-K defaultpriv=basic,net_privaddr,sys_resource,-prov_info,-file_link_any ds
```

2. Assign a password.

```
# passwd ds
```

3. For each administrator who is allowed to manage the Directory Server, assign the role with the `usermod` command. For example, to give someone with a user name of “john” the ability to assume the `ds` role, issue the following command:

```
# usermod -R ds john
```

If a user is already a member of one or more roles, then the entire list of existing roles, separated by commas, must also be provided or the user will be removed from those roles. For example, if the root account is also a role and the user “john” is also a member of that role, then the command would be:

```
# usermod -R root,ds john
```

4. Log in using a normal user account and then use the `bin/su` command to assume the role created for the Directory Server. You cannot log directly into a system as a role. Only users that have been explicitly assigned to a role will be allowed to assume it.

Chapter

3

Installing the Server

After you have prepared your hardware and software system based on the instructions in Chapter 2, you can begin the setup process using of the Ping Identity Directory Server's easy-to-use installation modes.

This chapter presents the various installation options and procedures available to the administrator.

Topics:

- [*Getting the Installation Packages*](#)
- [*About the Layout of the Directory Server Folders*](#)
- [*About the Server Installation Modes*](#)
- [*Before You Begin*](#)
- [*Setting Up the Directory Server in Interactive Mode*](#)
- [*Installing the Directory Server in Non-Interactive Mode*](#)
- [*Installing a Lightweight Server*](#)
- [*Running the Status Tool*](#)
- [*Where To Go From Here*](#)
- [*Working with Multiple Backends*](#)
- [*Importing Data*](#)
- [*Running the Server*](#)
- [*Stopping the Directory Server*](#)
- [*Uninstalling the Server*](#)

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from PingData and unpack it in a folder of your choice. The release bundle contains the Directory Server code, tools, and package documentation.

To Unpack the Build Distribution

1. Download the latest zip distribution of the Directory Server software.
2. Unzip the compressed zip archive file in a directory of your choice.

```
$ unzip PingDirectory-<version>.zip
```

You can now set up the Directory Server.

About the RPM Package

PingData supports the Ping Identity Directory Server release bundle in an RPM Package Manager (RPM) package for customers who require it. By default, the RPM unpacks the code at `/opt/ping-identity/ds/PingDirectory`, after which you can run the `setup` command to install the server at that location.

If the RPM install fails for any reason, you can perform an RPM erase if the RPM database entry was created and manually remove the target RPM install directory (e.g., `/opt/ping-identity/ds/PingDirectory` by default). You can install the package again once the system is ready.

To Install the RPM Package

1. Download the latest RPM distribution of the Directory Server software.
2. Unpack the build using the `rpm` command with the `--install` option. By default, the build is unpacked to `/opt/ping-identity/ds/PingDirectory`. If you want to place the build at another location, use the `--prefix` option and specify the file path of your choice.

```
$ rpm --install PingDirectory-<version>.rpm
```

3. From `/opt/ping-identity/ds/PingDirectory/PingDirectory`, run the `setup` command to install the server on the machine.

About the Layout of the Directory Server Folders

Once you have unzipped the Directory Server distribution file, you will see the following folders and command-line utilities, shown in the table below.

Table 1: Layout of the Directory Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Server.
README	README file that describes the steps to set up and start the Directory Server.
bak	Stores the physical backup files used with the <code>backup</code> command-line tool.
bat	Stores Windows-based command-line tools for the Directory Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the Data Metrics Server.
config	Stores the configuration files for the backends (<code>admin</code> , <code>config</code>) as well as the directories for messages, schema, tools, and updates.
db	Stores the Oracle Berkeley Java Edition database files for the Directory Server.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notice	Stores any legal notices for dependent software used with the Directory Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the Data Metrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The <code>revert-update</code> tool for UNIX/Linux systems.
revert-update.bat	The <code>revert-update</code> tool for Windows systems.
setup	The <code>setup</code> tool for UNIX/Linux systems.
setup.bat	The <code>setup</code> tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The <code>uninstall</code> tool for UNIX/Linux systems.
uninstall.bat	The <code>uninstall</code> tool for Windows systems.
update	The <code>update</code> tool for UNIX/Linux systems.
update.bat	The <code>update</code> tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

About the Server Installation Modes

One of the strengths of the Ping Identity Directory Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to quickly install and configure a stand-alone Directory Server instance.

To install a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode.** Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command.
- **Non-Interactive Command-Line Mode.** Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the `--no-prompt` option as well as the other arguments required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Server will run.

Before You Begin

After you have unzipped the Directory Server ZIP file, you may want to carry out the following functions depending on your deployment requirements:

- **Custom Schema Elements.** If your deployment uses custom schema elements in a custom schema file (for example, `98-schema.ldif`), you may do one of the following:
 - Copy your custom schema file to the `config/schema` directory before running `setup`.
 - Copy your custom schema file to the `config/schema` directory after `setup` and re-start the server. If replication is enabled, the restart will result in the schema replicating to other servers in the replication topology.
 - Use the **Schema Editor** after `setup`. If replication is enabled, schema definitions added through the **Schema Editor** will replicate to all servers in the replication topology without the need for a server restart.
- **Certificates.** If you are setting up a new machine instance, copy your keystore and truststore files to the `<server-root>/config` directory prior to running `setup`. The keystore and truststore passwords can be placed, in clear text, in corresponding `keystore.pin` and `truststore.pin` files in `<server-root>/config`.
- **Locations.** Location names are used to define a grouping of Ping Identity Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.
- **Validate ACIs.** Many directory servers allow for less restrictive application of its access control instructions (ACIs), so that they accept invalid ACIs. For example, if a Sun/Oracle server encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the Ping Identity Directory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can

cause problems when migrating data with existing access control rules to an Ping Identity Directory Server. If you are migrating from a Sun/Oracle deployment to an Ping Identity Directory Server, the Ping Identity Directory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. For more information, see [Validating ACIs Before Migrating Data](#).

Important:



Each Server Deployment Requires an Execution of Setup - Duplicating a Server-root is not Supported. The installation of the server does not write or require any data outside of the server-root directory. After executing `setup`, copying the server-root to another location or system, in order to *duplicate* the installation, is not a supported method of deployment. The server-root can be moved to another host or disk location if a host or file system change is needed.

Setting Up the Directory Server in Interactive Mode

The `setup` tool also provides an interactive text-based command-line interface to set up a Directory Server instance.

To Install the Directory Server in Interactive Mode

1. Unzip the distribution ZIP file, review [Before You Begin](#), and then go to the server root directory. Use the `setup` utility with the `--cli` option to install the server in interactive mode.

```
$ ./setup --cli
```

If the `JAVA_HOME` environment variable is set to an older version of Java, explicitly specify the path to the Java JDK installation during the setup process. Either set the `JAVA_HOME` environment variable with the Java JDK path or execute the `setup` command in a modified Java environment using the `env` command.

```
$ env JAVA_HOME=/ds/java ./setup --cli
```

2. Read the PingData End-User License Agreement, and type `yes` to continue.
3. Enter the fully qualified host name or IP address of the local host, or press **Enter** to accept the default.
4. Enter the root user DN, or press **Enter** to accept the default (`cn=Directory Manager`).
5. Enter and confirm the root user password.

6. Press **Enter** to enable the PingData services (Configuration, Documentation, and SCIM) and Administrative Console over HTTPS. After setup, individual services and applications can be enabled or disabled by configuring the HTTPS Connection Handler.
7. Enter the port on which the Directory Server will accept connections from HTTPS clients, or press **Enter** to accept the default.
8. Enter the port on which the Directory Server will accept connections from LDAP clients, or press **Enter** to accept the default.
9. The next two options enable using LDAPS or StartTLS. Type `no` to use a standard LDAP connection, or accept the default (`yes`) to enable both. Enabling LDAPS configures the LDAPS Connection Handler to allow SSL over its client connections. Enabling StartTLS configures the LDAP Connection Handler to allow StartTLS.
10. Select the certificate option for this server:
 - Generate a self-signed certificate for testing purposes only.
 - To use an existing certificate using a Java Keystore, enter the keystore path and keystore PIN.
 - To use an existing certificate using use a PKCS#12 keystore, enter the keystore path and the keystore PIN.
 - To use the PKCS#11 token, enter only the keystore PIN.
11. Type the base DN for the data, or accept the default base DN of `dc=example,dc=com`.
12. Choose an option to generate and import sample data. Type the desired number of entries, or press `Enter` to accept the default number (10000). This option is used for quick evaluation of the Directory Server.

See [Initializing Data onto the Server](#) if you want to use other options to initialize the server.
13. Choose the option to tune the amount of memory that will be consumed by the Directory Server and its tools.
14. Press `Enter` to prime or preload the database cache at startup prior to accepting client connections.

Priming the cache can increase the startup time for the Directory Server but provides optimum performance once startup has completed. This option is best used for strict throughput or response time performance requirements, or if other replicas in a replication topology can accept traffic while this Directory Server instance is starting. Priming the cache also helps determine the recommended JVM option, `CMSInitiatingOccupancyFraction`, when a Java garbage collection pause occurs. See [JVM Garbage Collection Using CMS](#).
15. Press **Enter** to accept the default (`yes`) to start the Directory Server after the configuration has completed. To configure additional settings or import data, type `no` to keep the server in shutdown mode.
16. Choose an option to continue server set up.

17. On the **Setup Summary** page, confirm the configuration. Press **Enter** to accept the default (set up with the parameters given), enter the option to repeat the installation process, or enter the option to cancel the setup completely.

Installing the Directory Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. Non-interactive mode is useful when setting up production or QA servers with specific configuration requirements.

The non-interactive command-line mode requires that all mandatory options be present for each command call. If there are missing or incorrect arguments, the `setup` tool fails and aborts the process. You must also use a `--no-prompt` option to suppress interactive output, except for errors, when running in non-interactive mode. Additionally, you must also use the `--acceptLicense` option and specify the port using the `--ldapPort` or `--ldapsPort` option. If neither option is specified, an error message is displayed. To view the license, run `bin/review-license` command.

To automatically tune the JVM to use maximum memory, use the `--aggressiveJVMTuning` and `--maxHeapSize {memory}` options. To preload the database at startup, use the `--primeDB` option.

To configure a deployment using a truststore, see [Installing the Directory Server in Non-Interactive Mode with a Truststore](#).

To see a description of the available command-line options for the `setup` tool, use `setup --help`.

To Install the Directory Server in Non-Interactive Mode

The following procedure shows how to install a Directory Server in a production or QA environment with no security enabled.

- Unzip the distribution ZIP file, review “Before You Begin”, and then use `setup` with the `--cli` and `--no-prompt` options for non-interactive mode from the `<server-root>` directory. The following command uses the default root user DN (`cn=Directory Manager`) with the specified `--rootUserPassword` option. You must include the `--acceptLicense` option or the setup will generate an error message.

```
$ ./setup --cli --no-prompt --rootUserPassword "password" \
  --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389
```

To Install the Directory Server in Non-Interactive Mode with a Truststore

You can set up the Directory Server using an existing truststore for secure communication. This section assumes that you have an existing keystore and truststore with trusted certificates.

- Unzip the distribution ZIP file, review [Before You Begin](#), and then, from the server root directory, use `setup` with the `--cli` and `--no-prompt` options for non-interactive mode.

The following example enables security using both SSL and StartTLS. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The `userRoot` database contents will remain empty and the base DN entry will not be created.

```
$ ./setup --cli --no-prompt --rootUserPassword "password" \  
--baseDN "dc=example,dc=com" --ldapPort 389 --enableStartTLS \  
--ldapsPort 636 --useJavaKeystore config/keystore.jks \  
--keyStorePasswordFile config/keystore.pin \  
--certNickName server-cert --useJavaTrustStore config/truststore.jks \  
--acceptLicense
```

The password to the private key with the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined with the Administrative Console or the `dsconfig` tool by editing the Trust Manager Provider standard configuration object.

Installing a Lightweight Server

Users who want to demo or test a lightweight version of the Directory Server on a memory-restricted machine can do so by removing all unused or unneeded configuration objects. All configuration entries, whether enabled or not, take up some amount of memory to hold the definition and listeners that will be notified of changes to those objects.

The configuration framework will not allow you to remove objects that are referenced, and in some cases if you have one configuration object referencing another but really do not need it, then you will first need to remove the reference to it. If you try to remove a configuration object that is referenced, both `dsconfig` and the Administrative Console should prevent you from removing it and will tell you what still references it.

Depending on your test configuration, some example configuration changes that can be made are as follows:

- **Reduce the number of worker threads.** Each thread has a stack associated with it, and that consumes memory. If you're running a bare-bones server, then you probably do not have enough load to require a lot of worker threads.

```
$ bin/dsconfig set-work-queue-prop \  
--set num-worker-threads:8 \  
--set num-administrative-session-worker-threads:4 \  
--set max-work-queue-capacity:100
```

- **Reduce the percentage of JVM memory used for the JE database cache.** When you have a memory-constrained environment, you want to ensure that as much of the memory that is there is available for use during processing and not tied up caching database contents.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:5
```

- **Disable the Dictionary Password Validator.** The Dictionary Password Validator takes a lot of memory to hold its dictionary. Disabling it will free up some memory. You can delete the other password validators if not needed, such as Attribute Value, Character Set, Length-based, Repeated Characters, Similarity-based, or Unique Characters Password Validator.

```
$ bin/dsconfig delete-password-validator --validator-name Dictionary
```

- **Remove non-essential schema files.** Although not recommended for production deployments, some candidates that you can remove are the following: 03-rfc2713.ldif, 03-rfc2714.ldif, 03-rfc2739.ldif, 03-rfc2926.ldif, 03-rfc2985.ldif, 03-rfc3712.ldif, 03-uddiv3.ldif.

There are other items that can be removed, depending on your desired configuration. Contact your authorized support provider for assistance.

Running the Status Tool

The Directory Server provides a `status` tool that outputs the current state of the server as well as other information, such as server version, JE Environment statistics, Operation Processing Times, Work Queue, and Administrative Alerts. The `status` tool is located in the `bin` directory (UNIX, Linux) or the `bat` directory (Windows).

To Run the Status Tool

- Run the `status` command on the command line. The following command displays the current Directory Server status and limits the number of viewable alerts in the last 48 hours. It provides the current state of each connection handler, data sources, JE environment statistics, processing times by operation type and current state of the work queue.

```
$ bin/status --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret

--- Server Status ---
Server Run Status:   Started 28/Mar/2012:10:47:17.000 -0500
Operational Status: Available
Open Connections:   13
Max Connections:    13
Total Connections:  50

--- Server Details ---
Host Name:          server1.example.com
Administrative Users: cn=Directory Manager
Installation Path:  PingDirectory
Server Version:     Ping Identity Directory Server 6.0.1.0
Java Version:       jdk-7u9

--- Connection Handlers ---
Address:Port : Protocol : State
-----:-----:-----:-----
0.0.0.0:1389 : LDAP   : Enabled
0.0.0.0:1689 : JMX    : Disabled
0.0.0.0:636  : LDAPS  : Disabled

--- Data Sources ---
Base DN:           dc=example,dc=com
Backend ID:        userRoot
Entries:           2003
Replication:       Enable
Replication Backlog: 0
Age of Oldest Backlog Change: not available

--- JE Environment ---
ID                : Cache Full : Cache   : On-Disk : Alert
-----:-----:-----:-----:-----
replicationChanges : 6 %        : 328.8 kb : 30.4 kb : None
userRoot           : 9 %        : 6.2mb   : 146.6mb : None

--- Operation Processing Time ---
Op Type   : Total Ops : Avg Resp Time (ms)
-----:-----:-----
Add       : 0         : 0.0
```

```

Bind      : 0          : 0.0
Compare  : 0          : 0.0
Delete   : 0          : 0.0
Modify   : 2788567    : 0.921
Modify   : 0          : 0
DN       : 2267266    : 0.242
Search   : 5055833    : 0.616
All

      --- Work Queue ---
      : Recent : Average : Maximum
-----:-----:-----:-----
Queue Size : 4      : 0      : 10
% Busy     : 26     : 5      : 100

      --- Administrative Alerts ---
Severity : Time           : Message
-----:-----:-----
Info     : 28/Mar/2012 10:47:17 -0500 : The Directory Server has started successfully
Info     : 28/Mar/2012 10:47:14 -0500 : The Directory Server is starting
Info     : 28/Mar/2012 10:44:22 -0500 : The Directory Server has started successfully
Info     : 28/Mar/2012 10:44:18 -0500 : The Directory Server is starting

Shown are alerts of type [Info,Warning,Error,Fatal] from the past 48 hours Use the
--maxAlerts and/or --severity options to filter this list

```



Note: By default, the `status` command displays the alerts generated in the last 48 hours. You can limit the number of alerts by using the `--maxAlerts` option.

Where To Go From Here

After you have set up your Directory Server instance, you can configure any specific server settings, import your user database, or run initial performance tests to optimize your server's throughput.

- **Log into the Administrative Console.** Become familiar with configuration options through the Administrative Console interface. The URL is based on the hostname and HTTPS port specified during installation, such as `https://hostname.com:443/console`.
- **Apply Server Configurations.** Apply your server configuration changes individually or using a `dsconfig` batch file. The batch file defines the Directory Server configuration tool, `dsconfig`, commands necessary to configure your server instance. For more information on using batch files, see [Using dsconfig in Batch Mode](#).

If you are migrating from a Sun Java System 5.x, 6.x, 7.x directory server, you can use the `bin/migrate-sun-ds-config` command to migrate your configuration settings to this newly installed server instance.

- **Import Data.** Import user data using the `import-ldif` tool. The import serves as an initial test of the schema settings.

```
$ bin/import-ldif --backendID userRoot --ldifFile ../user-data.ldif
```

- **Run Performance Tests.** The Directory Server provides two tools for functional performance testing using in-house LDAP clients that accesses the server directly: `searchrate` (tests search performance) and `modrate` (tests modification performance):


```
$ bin/searchrate --baseDN "dc=example,dc=com" --scope sub \  
  --filter "(uid=user.[0-1999])" --attribute givenName --attribute sn \  
  --attribute mail --numThreads 10  
  
$ bin/modrate --entryDN "uid=user.[0-1999],ou=People,dc=example,dc=com" \  
  --attribute description --valueLength 12 --numThreads 10
```

Working with Multiple Backends

You can create multiple local database backends, each containing one or more different base DNs. There should be at most one replicating domain on each local database backend. The replication domain should not span multiple local database backends. The typical entry-balancing configuration involves two local database backends: one backend to serve the global domain data that resides above the entry-balancing point and a backend that is defined with the entry-balancing point as the base DN, such as `ou=people,dc=example,dc=com`.

With multiple local database backends configured, the data existing with each backend can be managed independently. In addition, separate index settings are applied to each local database backend.

When creating multiple database backends, consider the following:

- No two backends may have the same base DN.
- If any base DN for a given backend is subordinate to a base DN on another backend, then all base DNs on that backend must be subordinate to the base DN of the other backend.
- The total of all `db-cache-percent` values should be no more than 65-70% in most cases and should never be configured to exceed 100%.

Importing Data

After installation, the database, such as `userRoot`, will need to have data imported. For a server to be added to a replicating set, the database will be imported as part of the `dsreplication initialize` operation, which is performed after `dsreplication enable`. A server that will not be added to a replicating set, or the first server of a future replicating set, should have data imported with the `bin/import-ldif` tool. See Chapter 8, *Importing and Exporting Data*, for more information about the `bin/import-ldif` tool.

Generating Sample Data

The Ping Identity Directory Server provides LDIF templates that can be used to generate sample entries to initialize your server. You can generate the sample data with the `make-ldif` utility together with template files that come bundled with the ZIP build, or you can use template files that you create yourself. The templates create sequential entries that are convenient for testing the Ping Identity Directory Server with a range of dataset sizes. The Directory Server templates are located in the `config/MakeLDIF`.

To randomize the data, the `make-ldif` command has a `--randomSeed` option that can be used to seed the random number generator. If this option is used with the same seed value, the template will always generate exactly the same LDIF file.

The sample data templates generate a dataset with basic access control privileges that grants anonymous read access to anyone, grants users the ability to modify their own accounts, and grants the Admin account full privileges. The templates also include the `uid=admin` and `ou=People` entries necessary for a complete dataset. You can bypass the `make-ldif` command entirely and use the `--templateFile` option with the `import-ldif` tool.

- Use the `make-ldif` command to generate sample data. The command generates 10,000 sample entries and writes them to an output file, `data.ldif`. The random seed generator is set to 0.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10k.template \  
--ldifFile /path/to/data.ldif --randomSeed 0
```

To Import Data on the Directory Server Using Offline Import

1. Create an LDIF file that contains entries, or locate an existing file.

The `import-ldif` tool requires an LDIF file, which conforms to standard LDIF syntax without change records. This means the `changeType` attribute is not allowed in the input LDIF. For information on adding entries to the Directory Server, see [Managing Entries](#).

2. Stop the Directory Server.

3. Use the offline `import-ldif` to import data from an LDIF file to the Directory Server. For assistance with the list of options, run `import-ldif --help`.

In the following example, the data is imported from the `data.ldif` file to the `userRoot` backend. If any entry is rejected due to a schema violation, then the entry and the reason for the rejection is written to the `rejects.ldif` file. Skipped entries, written to `skipped.ldif`, occur if an entry cannot be placed under a branch node in the DIT or if exclusion filtering is used (`--excludeBranch`, `--excludeAttribute`, or `--excludeFilter`). The `--overwrite` option instructs `import-ldif` to overwrite existing skipped and rejected files. The `--overwriteExistingEntries` option indicates that any existing data in the backend should be overwritten. Finally, the `--stripTrailingSpaces` option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif --rejectFile \  
rejects.ldif --skipFile skipped.ldif --overwrite --overwriteExistingEntries -- \  
stripTrailingSpaces
```

4. Re-start the Directory Server.

Running the Server

To start the Directory Server, run the `bin/start-ds` command on UNIX or Linux systems (an analogous command is in the `bat` folder on Microsoft Windows systems). The `bin/start-ds` command starts the Directory Server as a background process when no options are specified.

To run the Directory Server as a foreground process, use the `bin/start-ds` command with the `--nodetach` option.

To Start the Directory Server

Use `bin/start-ds` to start the server.

```
$ bin/start-ds
```

To Run the Server as a Foreground Process

1. Enter `bin/start-ds` with the `--nodetach` option to launch the Directory Server as a foreground process.

```
$ bin/start-ds --nodetach
```

2. You can stop the Directory Server by pressing `CNTRL+C` in the terminal window where the server is running or by running the `bin/stop-ds` command from another window.

To Start the Server at Boot Time

By default, the Ping Identity Directory Server does not start automatically when the system is booted. Instead, you must manually start it with the `bin/start-ds` command. To configure the Directory Server to start automatically when the system boots, use the `create-rc-script` utility to create a run control (RC) script, or create the script manually.

1. Create the startup script.

```
$ bin/create-rc-script --outputFile PingDirectory.sh --userName ds
```

2. As a root user, move the generated `PingDirectory.sh` script into the `/etc/init.d` directory and create symlinks to it from the `/etc/rc3.d` directory (starting with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

```
# mv PingDirectory.sh /etc/init.d/  
# ln -s /etc/init.d/PingDirectory.sh /etc/rc3.d/S50-boot-ds.sh  
# ln -s /etc/init.d/PingDirectory.sh /etc/rc0.d/K50-boot-ds.sh
```

Some Linux implementations may not like the “-” in the scripts. If your scripts do not work, try renaming the scripts without the dashes. You can also try symlinking the `S50*` file into the `/etc/rc3.d` or the `/etc/rc0.d` directory or both, based on whatever runlevel the server enters when it starts. Some Linux systems do not even use `init.d`-style startup scripts, so depending on whatever flavor of Linux you are using you might have to put the script somewhere else or use some other mechanism for having it launched at startup.

3. Log out as root, and re-assume the `ds` role if you are on a Solaris system.

Logging into the Administrative Console

After the server is installed, access the Administrative Console, `https://server-name/console/login`, to verify the configuration and manage the server. To log into the Administrative Console, use the initial root user DN specified during setup (by default `cn=Directory Manager`).

The `dsconfig` command or the Administrative Console can be used to create additional root DN users in `cn=Root DNs,cn=config`. These new users require the fully qualified DN as the login name, such as `cn=new-admin,cn=Root DNs,cn=config`. To use a simple user name (with out the `cn=` prefix) for logging into the Administrative Console, the root DN user must have the `alternate-bind-dn` attribute configured with an alternate name, such as "admin."

By default the link to the Administrative Console is `https://server-name/console/login`.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

Stopping the Directory Server

The Directory Server provides a simple shutdown script, `bin/stop-ds`, to stop the server. You can run it manually from the command line or within a script.

If the Directory Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the `vmstat` command and watch the values in the "free" column increase until all memory held by the Directory Server is released back to the system.

You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Server

- Use the `bin/stop-ds` tool to shut down the server.

```
$ bin/stop-ds
```

To Schedule a Server Shutdown

- Use the `bin/stop-ds` tool with the `--stopTime YYYYMMDDhhmmss` option to schedule a server shutdown.

The Directory Server schedules the shutdown and sends a notification to the `server.out` log file. The following example sets up a shutdown task that is scheduled to be processed on June 6, 2012 at 8:45 A.M. CDT. The server uses the UTC time format if the provided timestamp includes a trailing “Z”, for example, 20120606134500Z. The command also uses the `--stopReason` option that writes the reason for the shut down to the logs.

```
$ bin/stop-ds --stopTime 20120606134500Z --port 1389 \  
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \  
--stopReason "Scheduled offline maintenance"
```

To Restart the Server

Re-start the Directory Server using the `bin/stop-ds` command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again.

- Go to the server root directory, and run the `bin/stop-ds` command with the `-R` or `--restart` options.

```
$ bin/stop-ds --restart
```

Uninstalling the Server

The Directory Server provides an `uninstall` command-line utility for quick and easy removal of the code base.

To uninstall a server instance, run the `setup` tool in one of the following modes: interactive command-line, or non-interactive command-line mode.

- **Interactive Command-Line Mode.** Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the `bin/uninstall` command with the `--cli` option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode.** Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the `bin/uninstall` command with the `--no-prompt` option.



Note: For stand-alone installations with a single Directory Server instance, you can also manually remove the Directory Server by stopping the server and recursively deleting the directory and subdirectories. For example:

```
$ rm -rf /ds/PingDirectory
```

To Uninstall the Server in Interactive Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Server.

1. From the server root directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press **Enter** to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?

1) Remove all components
2) Select the components to be removed

q) quit
Enter choice [1]:
```

3. For each type of server component, press **Enter** to remove them or type `no` to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.
5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.
6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

```
How do you want to trust the server certificate for the Directory Server
on server.example.com:389?

1) Automatically trust
2) Use a trust store
3) Manually validate

Enter choice [3]:
```

7. If your Directory Server is running, the server is shutdown before continuing the uninstall process. The `uninstall` processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

To Uninstall the Server in Non-Interactive Mode

The `uninstall` utility provides a non-interactive method to enter the command with the `--no-prompt` option. Another useful argument is the `--forceOnError` option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the `--forceOnError` option is not used, the command will fail and abort.

1. From the server root directory, run `uninstall` tool with the `--remove-all` option to remove all of the Directory Server's libraries. The `--quiet` option suppresses output information and is optional. The following command assumes that the Directory Server is stand-alone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

From the server root directory, run `uninstall` with the `--backup-files` option to remove the Directory Server's backup files. Use the `--help` or `-H` option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```


Chapter

4 Upgrading the Server

PingData issues software release builds periodically with new features, enhancements, and fixes for improved server performance. Administrators can use the Directory Server's update utility to upgrade the current server code version.

This chapter presents some update scenarios and their implications that you should consider when upgrading your server code.

Topics:

- [*Upgrade Overview and Considerations*](#)
- [*To Upgrade the Directory Server*](#)
- [*To Upgrade the RPM Package*](#)
- [*Reverting an Update*](#)
- [*Configure SCIM After Upgrade*](#)

Upgrade Overview and Considerations

The upgrade process involves downloading and unzipping a new version of the Directory Server ZIP file on the server to be updated, and running the `update` utility with the `--serverRoot` or `-R` option value from the new root server pointing to the installation to be upgraded.

Consider the following when upgrading replicating servers:

- Upgrade affects only the server being upgraded. The process does not alter the configuration of other servers.
- The `update` tool will verify that the version of Java that is installed meets the new server requirements. To simplify the process, install the version of Java that is supported by the new server before running the tool.
- To be safe, backup the user data (`userRoot`) before an upgrade. Restoring from a backup could be necessary if all other servers in the replication topology have been upgraded and a database or encoding change in the new server version prevents the database from being used with the older server version. The `update` and `revert-update` utilities will issue a warning when this is the case.
- Temporarily raise the replication purge delay for all servers in the topology to cover the expected downtime for maintenance. This will result in a temporary increase in disk usage for the replicationChanges database stored in `<server-root>/changeLogDb`.
- Replication does not need to be disabled on a server before an upgrade.
- Make sure upgraded servers are working as expected before upgrading the last server in the topology
- Enable new features after all replicating servers are upgraded.

To Upgrade the Directory Server

Perform an upgrade with the following steps.

1. Download and unzip the new version of the Directory Server in a location outside the existing server's installation. For these steps, assume the existing server installation is in `/prod/PingDirectory` and the new server version is unzipped into `/home/stage/PingDirectory`.
2. Run the `update` tool provided with the new server package to update the existing Directory Server. The update tool may prompt for confirmation on server configuration changes if it detects customization.

```
$ /home/staging/PingDirectory/update --serverRoot /prod/PingDirectory
```

To Upgrade the RPM Package

If the Linux RPM package was used to install the Directory Server, the following should be performed to upgrade the server.

- Assume that the new RPM package, PingDirectory-<new-version>.rpm, is placed in the server root directory. From the server root directory, run the `rpm` command with the `--upgrade` option.

```
$ rpm --upgrade PingDirectory-<new-version>.rpm
```

The RPM package does not support a revert option once the build is upgraded.

The upgrade history is written to `/opt/ping-identity/ds/PingDirectory/PingDirectory/history/<timestamp>/update.log`.

Reverting an Update

An installation can be reverted (one level back) using the `revert-update` tool. The `revert-update` tool will place the server's binaries and configuration back to its prior state. If multiple updates have been performed, the tool can be run multiple times to revert to each prior update sequentially, but only one level at a time.

1. Stop the server and run the `revert-update` command:

```
$ bin/stop-ds
$ ./revert-update
```

2. Before starting the server, make sure there were no warnings, such as:

Warning: Additional steps must be taken once this revert has completed. The current server uses an on disk database format that is not compatible with the server version being reverted to. After performing the revert, compatible databases must be put in place before restarting the server. Different steps must be taken to restore the Local DB Backend, Replication Changelog, and LDAP Changelog databases.

This type of message indicates that the userRoot, replicationChange and changelog (if enabled) databases need to be restored before starting the server. These databases can be backed up from another server of the same version in the replication topology and restored on the newly reverted server. For example, from another server in the topology, perform the following:

```
$ bin/backup --backendID userRoot --backupDirectory /backups/userRoot
$ bin/backup --backendID replicationChanges --backupDirectory /backups/
replicationChanges
$ bin/backup --backendID changelog --backupDirectory /backups/changelog
```

Transfer the backups to the reverted server and restore each database with the following commands:

```
$ bin/restore --backupDirectory /transferred/userRoot
$ bin/restore --backupDirectory /transferred/changelog
$ bin/restore --backupDirectory /transferred/replicationChanges
```

3. After all databases have been restored, start the server with the following command:

```
$ bin/start-ds
```

Configure SCIM After Upgrade

Modifications in SCIM PATCH are mapped directly to LDAP modifications to use the matching rules configured in the Directory Server, when matching deleted values. Since the SCIM PATCH is now applied by the Directory Server, the Permissive Modify Request Control (1.2.840.113556.1.4.1413) is now required by the SCIM component. This ensures that adding an existing value or deleting a non-existent value in the PATCH request will not generate an error. This affects upgrades from server versions prior to 5.0.0.

To continue using the SCIM component after an upgrade, access controls and configuration must be updated to allow access to the Permissive Modify Request Control. Run the `dsconfig` commands to update these components:

```
$ dsconfig set-access-control-handler-prop \  
  --remove 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473 ||  
  1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12")(version 3.0;acl  
  "Authenticated access to controls used by the SCIM servlet extension"; allow (all)  
  userdn="ldap:///all";)'
```

```
$ dsconfig set-access-control-handler-prop \  
  --add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473  
  || 1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9 || 1.3.6.1.1.12 ||  
  1.2.840.113556.1.4.1413")(version 3.0;acl "Authenticated access to controls used by the  
  SCIM servlet extension"; allow (all) userdn="ldap:///all";)'
```

Chapter

5

Tuning the Server

The Ping Identity Directory Server's installation process automatically determines the optimal Java Virtual Machine (JVM) settings based on calculations of the machine running setup. While the majority of the default configuration and JVM settings are suitable for most deployments, it is not uncommon in high performance environments to make slight adjustments to the Directory Server's JVM settings as well as performance and resource-related configuration changes with the `dsconfig` tool.

This chapter provides guidance for tuning the Directory Server and its tools for both optimum performance with regard to throughput and disk space usage. This chapter presents the following topics:

Topics:

- [*About Minimizing Disk Access*](#)
- [*Memory Allocation and Database Cache*](#)
- [*Database Preloading*](#)
- [*Databases on Storage Area Networks, Network-Attached Storage, or running in Virtualized Environments*](#)
- [*Database Cleaner*](#)
- [*Compacting Common Parent DNs*](#)
- [*Import Thread Count*](#)
- [*JVM Properties for Server and Command-Line Tools*](#)
- [*JVM Garbage Collection Using CMS*](#)
- [*Tuning For Disk-Bound Deployments*](#)
- [*Uncached Attributes and Entries*](#)

About Minimizing Disk Access

Most critical to directory server performance is minimizing disk access. Defining a JVM heap size that can contain the entire contents of the database cache in memory is essential to minimizing read operations from disk and achieving optimal performance. It is also important to understand that the database on-disk is comprised of transaction log files, which are only appended to. After an initial database import, the size on-disk will grow by a factor of at least 25% as inactive records accumulate within the transaction logs. Therefore, during normal operation, the on-disk size of the database transaction logs do not represent the memory needed to cache the database.

Another consideration is to minimize the size of the database based on the known characteristics of your data. Minimizing the size of the database not only reduces hard disk requirements but also reduces the memory requirements for the database cache. The Directory Server has the capability to automatically compact common parent DNs, which is an example of optimizing the database size based on known characteristics of the data.

Another consideration is to consider the write load on your server and its affect on the database. While write operations will always require an associated write to disk, an environment that sustains a high load of write operations may consider tuning the background database cleaner to minimize the size of the database on disk.

Memory Allocation and Database Cache

The Directory Server's optimal performance is dependent on the proper allocation of memory to the JVM heap, the number of processor cores in the system, and the correct combination of JVM options for optimized garbage collection. The `setup` tool for the Directory Server automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system. However, in most production deployments, additional tuning may be required to meet the performance objectives for your system.

Most often, directory server performance tuning can be accomplished by adjusting a few settings. Tuning these settings, which include both JVM and configuration options, require an understanding of the JVM heap structure as well as the expected database usage. This section describes the basic components of the Directory Server footprint and logic behind the automated tuning of the `setup` tool.

Directory Server Process Memory

The Directory Server is comprised mostly of a JVM heap and a marginal amount of memory allocated by the JVM's execution of native code. While we frequently refer to the JVM Heap as the maximum memory consumed by the Directory Server, the actual process size will be slightly larger than the `Xmx` value due to accumulation of small chunks of native code that Java requires for things, such as SSL sockets.

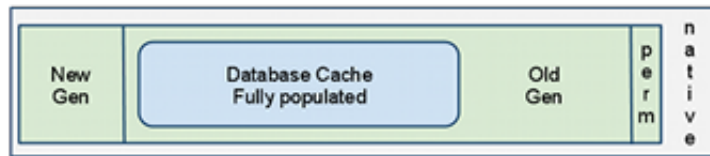


Figure 2: JDK Heap Structure

Within the JVM Heap, the principal memory components are the New and Old Generations. The New Generation is a smaller area of memory where all data is initially allocated and is cleaned of garbage often. Any data that stays "alive" long enough will be promoted to the Old Generation for the longer term. The Old Generation is where the database cache will eventually reside. The Old Generation size is computed from the leftover heap after defining the `MaxHeapSize` and New Generation sizes; therefore, it is not explicitly stated in the JVM options. A typical set of Generation definitions for the JVM is as follows, where `mx` and `ms` values represent the heap size:

```
-Xmx16g -Xms16g -XX:MaxNewSize=2g -XX-NewSize=2g
```



Note: The `mx` and `ms` values should always be the same, and the `MaxNewSize/NewSize` values should be the same. This will help avoid negative changes in performance.

The `MaxNewSize/NewSize` values should never need to exceed 2g. The `setup` and `dsjavaproperties` tools set `MaxNewSize/NewSize` values based on the results of extensive performance testing, and should not need to be changed.

A Method for Determining Heap and Database Cache Size

The most straightforward approach to defining the proper memory allocation of the directory server components is to use the Directory Server `setup` command on hardware that represents the target production platform, especially with regard to process and memory, and the largest heap size that the `setup` tool will allow. After running `setup`, any schema and production database settings should be defined in preparation for the database import using the `import-ldif` tool.



Note: At the moment after an `import-ldif`, the database is at its most optimized state on disk with no inactive records. Over time, the on-disk representation of the database will grow as much as 25-50% as inactive records accumulate before being removed by the server's cleaner thread.

After the database is imported, the server should be started and a configuration change made to the backend. In this scenario, set the `prime-method` to "preload" on the userRoot backend configuration. Once the change is made, re-start the Directory Server and watch for a successful preload message at the end of startup. If preloading did not complete, the server should be stopped. The `start-ds.java-args` entry in the `config/java.properties` file should be

edited to use larger values for `-Xmx` and `-Xms` arguments. Then run the `bin/dsjavaproperties` command and restart the server. If preloading completed successfully, the database cache utilization percentage will be of interest. The `status` command will display something like the following:

```
--- JE Environment ---
ID      : Cache Full : Cache   : On-Disk : Alert
-----:-----:-----:-----:-----
userRoot : 30%       : 1.1 gb : 868.6mb : None
```

Looking at the above output and knowing that the database is fully loaded into cache, the 30% utilization is comfortable cushion for future database growth. In general, it is best to leave at least 10-20% cache headroom available.

During this scenario, it was clear from the `start-ds` output that the database primed completely and our interpretation of the `status` output was sound. To see the state of the database cache with more detail, perform an `ldapsearch` on the backend monitor.

In addition to the user configured backends, there may be backends for replication and changelog. The heap is shared among all backends. The amount allocated to each backed is calculated according to the procedure in the next section.

Automatic DB Cache Percentages

The setup process automatically tunes the percentage of the `db-cache-percent` property for the `userRoot` backend based on the maximum configured JVM heap size. This is only done for the `userRoot` backend during setup. Other backends created by the user are allocated 10%. The allocation can be changed if needed. When setting up the server, perform the following steps:

- Install the server with necessary memory. The server will autotune the size of the cache.
- Set the autotuned cache size to the limit for the combined cache sizes of all of the backends.
- Divide the server cache based on the expected size of the data in each backend.

Automatic Memory Allocation

If the Memory Tuning feature is enabled during setup, the `setup` algorithm determines the maximum JVM heap size based on the total amount of available system memory. If Memory Tuning is not selected, the server allocates a maximum JVM heap of 384 MB. The Directory Server also allows you to specify the maximum heap size during the setup process. You can enable Memory Tuning during the setup process by selecting the feature during the interactive command-line mode, adding the `--jvmTuningParameter` option using the `setup` tool in non-interactive command-line mode, or regenerating the java properties file with `bin/dsjavaproperties` and the `--jvmTuningParameter` options (seen in *JVM Properties for Server and Command-Line Tools* page 75).

If Memory Tuning is selected, the server can allocate the maximum JVM heap depending on the total system memory. The following table displays the automatically allocated maximum JVM heap memory based on available system memory.

Table 2: Allocated Max JVM Memory if Tuning is Enabled

Available Memory	Allocated JVM Memory
16 GB or more using a 64-bit JVM	The maximum JVM heap size will be set to 70% of total system memory. If the maximum JVM heap size is less than or equal to 128GB of memory (which should be the case for systems with up to 160 GB of memory), then the initial heap size will be set to equal the maximum heap size. If the maximum heap size is greater than 128GB, then the initial heap size will be set to 128GB to work around an apparent bug in the JVM.
6 GB#16 GB using a 64-bit JVM	total system memory - 4 GB
4 GB–6 GB using a 64-bit JVM	2 GB
2 GB–4 GB	512 MB
1 GB–2 GB	384 MB

Automatic Memory Allocation for the Command-Line Tools

At setup, the Directory Server automatically allocates memory to each command-line utility based upon the maximum JVM heap size. The server sets each command-line utility in the `config/java.properties` with `-Xmx/Xms` values depending on the expected memory needs of the tools. Because some tools can be invoked as a server task while the server is online, there are two definitions of the tool in the `config/java.properties` file: one with `.online` and one with `.offline` added to the name. The online invocations of the tools typically require minimal memory as the task is performed within the Directory Server's JVM. The offline invocations of the tools, for example, `import-ldif.offline` and `rebuild-index.offline`, can require the same amount of memory that is needed by the Directory Server.

Beyond the offline tool invocations, some tools, such as `ldap-diff` and `verify-index`, may need more than the minimal memory if large databases are involved. The table below lists the tools that are expected to have more than the minimal memory needs along with the rules for defining the default heap size.

Table 3: Default Memory Allocation to the Command-Line Tools

Command-Line Tools	Allocated JVM Memory
<code>start-ds</code> , <code>import-ldif (offline)</code> , <code>rebuild-index (offline)</code>	MaxHeapSize
<code>backup (offline)</code> , <code>dbtest export-ldif (offline)</code> , <code>ldap-diff</code> , <code>restore (offline)</code> , <code>scramble-ldif</code> , <code>summarize-access-log</code> , <code>verify-index</code>	If Max System Memory is: <ul style="list-style-type: none"> > Greater than or equal to 16 GB: set Heap to 3 GB > Greater than or equal to 8 GB: set Heap to 1 GB > Greater than or equal to 4 GB: set Heap to 512 MB > Under 4 GB: set Heap to 256 MB

Database Preloading

Key to Directory Server performance is the ability to maintain the database contents in the database cache within the JVM memory. With a properly sized database cache, a priming method of "preload" directs the server to load the database contents into memory at server startup before accepting the first client connection. The time needed to preload the database is proportional to the database size. To avoid priming, the server can be started with the `start-ds --skipPrime` command. If the priming method is `none` or the `--skipPrime` option is specified at startup, the database cache will slowly build as entries are accessed. This could take several days to reach optimal performance.

The "preload" priming method is suitable for nearly all Directory Server deployments. If the size of the database precludes storing the whole database in memory, there are priming alternatives for optimizing server performance. This type of deployment is considered disk-bound since the disk is accessed when processing most operations. See the section [Disk-Bound Deployments](#) for more information. The remaining priming options are applicable to these environments.

The Directory Server database `prime-method` property configures how the caches get primed, what gets primed (data, internal nodes, system indexes) and where it gets primed (database cache, filesystem cache, or both). The `prime-method` property is a multi-valued option that enables preloading the internal nodes into the database cache before the server starts, and then primes the values in the background by cursoring across the database. For more details, see the [Ping Identity Directory Server Configuration Reference](#).

The following is a summary of the priming methods:

- **Preload All Data.** Prime the contents of the backend into the database cache.
- **Preload Internal Nodes Only.** Prime only internal database structure information into the database cache, but do not prime any actual data. (This corresponds to the `cache-keys-only` cache-mode.)
- **Cursor Across Indexes.** Use the `cursor-across-indexes` property to iterate through backend contents. This is similar to (and may be slower than) using the preload mechanism, but it enables priming to happen in the background after the server has started. This is used when shorter start up times are desired, and the slower performance of an uncached database is acceptable until the database is primed.

Configuring Database Preloading

Use the `dsconfig` tool to set the database priming method. If multiple prime methods are used, the order in which they are specified in the configuration is the order in which they will be performed. Changing the preloading option requires re-starting the Directory Server. The following procedure shows how to configure database preloading.

To Configure Database Preloading

1. Set the prime method to "preload" to load the database contents from disk into memory when the server starts up. This eliminates the need for the server to gradually prime the database cache using client traffic, and ensures that the server has optimal performance when it starts to receive client connections.

```
$ bin/dsconfig set-backend-prop \  
  --backend-name userRoot \  
  --set prime-method:preload
```

2. Re-start the Directory Server to apply the changes using `bin/stop-ds` and then, `bin/start-ds`.

To Configure Multiple Preloading Methods

1. To achieve the benefits of preloading without delaying server startup, `prime-method` can be set to `preload-internal-nodes-only`, which caches all of the keys within the database but not the values. The database values themselves can be cached in the background once the server has been started with the `cursor-across-indexes` option.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --add prime-method:preload-internal-nodes-only \
  --add prime-method:cursor-across-indexes \
  --set background-prime:true
```

2. Re-start the Directory Server to apply the changes using `bin/stop-ds` and then, `bin/start-ds`.

To Configure System Index Preloading

1. Some environments have many indexes configured, though only a few are used for performance-sensitive traffic. In this case, server start up time can be reduced by only preloading the necessary indexes into the database at startup.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload \
  --set prime-all-indexes:false \
  --set system-index-to-prime:dn2id \
  --set system-index-to-prime:id2entry
```

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name mail \
  --set prime-index:true
```

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name uid \
  --set prime-index:true
```

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name entryUUID \
  --set prime-index:true
```

2. Restart the Directory Server to apply the changes using `bin/stop-ds` and then, `bin/start-ds`.

Databases on Storage Area Networks, Network-Attached Storage, or running in Virtualized Environments

There are several considerations when using network-based storage or storage abstracted by virtualization that are not issues when databases are stored on local disks. A data durability problem occurs when remote storage or the virtualization environment experiences service interruptions, ranging from connectivity loss to total failure from power loss. Data corruption can occur when the storage layer accepts data for writing that is not made durable before a crash occurs. In these cases, a database property can be set that reduces the likelihood of data loss and

data corruption. The database property `database-on-virtualized-or-network-storage` can be set on a per-backend environment basis to request all database writes to be written durably to the underlying storage.

There is a performance penalty when enabling this property, and in most cases, is not recommended except where network storage is unreliable. For network file systems, the benefits of faster recovery and less likelihood of data loss from unplanned events may outweigh the penalty. The exact overhead of enabling `database-on-virtualized-or-network-storage` will depend on the characteristics of the database, the host filesystem, storage array configuration, and network and virtualization input and output parameters. The write overhead penalty may be substantial for SAN environments. Incremental and full backup strategies should be used instead if performance is unacceptable.

To enable `database-on-virtualized-or-network-storage` for each applicable backend, use the following command as an example, which references the configuration for the `userRoot` backend:

```
$ bin/dsconfig set-global-configuration-prop \
  --set database-on-virtualized-or-network-storage:true
```

This should be set to `false` if the database is on a local disk.

Database Cleaner

Production environments that have a high volume of write operations may require cleaner thread tuning to control the on-disk database size as log files with inactive nodes wait to be cleaned and deleted. The Directory Server stores its Oracle[®] Berkeley DB Java Edition (JE) database files on-disk in the `db` directory. Each JE database log file is labelled `nnnnnnnnn.jdb`, where `nnnnnnnnn` is an 8-digit hexadecimal number that starts at `00000000` and is increased by 1 for each file written to disk. JE only appends data to the end of each file and does not overwrite any existing data. JE uses one or more cleaner threads that run in the background to compact the number of JE database (`db`) files.

The cleaner threads begin by scanning the records in each `db` file, starting with the file that contains the smallest number of active records. Next, the cleaner threads append any active records to the most recent database file. If a record is no longer active due to modifications or deletions, the cleaner threads leave it untouched. After the `db` file no longer has active records, the cleaner threads can either delete the file or rename the discarded file. Note that because of this approach to cleaning, the database size on-disk can temporarily increase when cleaning is being performed and files are waiting to be removed.

The Local DB Backend configuration object has two properties that control database cleaning: `db-cleaner-min-utilization` and `db-num-cleaner-threads`. The `db-cleaner-min-utilization` property determines, by percentage, when to begin cleaning out inactive records from the database files. By default, the property is set to 75, which indicates that database cleaning ensures that at least 75% of the total log file space is devoted to live data. Note that this property only affects the on-disk representation of the database and not the in-memory database cache—only live data is ever cached in memory.

The `db-num-cleaner-threads` property determines how many threads are configured for `db` cleaning. The default single cleaner thread is normally sufficient. However, environments with a

high volume of write traffic may need to increase this value to ensure that database cleaning can keep up.

If the number of database files grow beyond your expected guidelines or if the Directory Server is experiencing an increased number of update requests, you can increase the number of cleaner threads using the `dsconfig` tool (select **Backend** > **select advanced properties** > **db-num-cleaner-threads**).

Compacting Common Parent DNs

The Ping Identity Directory Server compacts entry DNs by tokenizing common parent DNs. Tokenizing the common parent DNs allows you to increase space usage efficiency when encoding entries for storage. The Directory Server automatically defines tokens for base DNs for the backend (for example, `dc=example,dc=com`). You can also define additional common base DNs that you want to tokenize. For example, use the following configuration to tokenize two branches, `ou=people,dc=example,dc=com` and `ou=customers,dc=example,dc=com`:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \  
  --add "compact-common-parent-dn:ou=people,dc=example,dc=com" \  
  --add "compact-common-parent-dn:ou=customers,dc=example,dc=com"
```

Import Thread Count

For most systems, the default setting of 16 threads is sufficient and provides good import performance. On some systems, increasing the import thread count may lead to improved import performance, while selecting a value that is too large can actually cause import performance to degrade. If minimizing LDIF import time is crucial to your deployment, you must determine the optimal number of import threads for your system, which is dependent on both the underlying system and the dataset being imported.

You can use the `dsconfig` command to set the number of import threads as follows:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set import-thread:24
```

JVM Properties for Server and Command-Line Tools

The Directory Server and tools refer to the `config/java.properties` file for JVM options that include important memory settings. The `java.properties` file sets the default Java arguments for the Directory Server and each command-line utility including the default `JAVA_HOME` path.

The `java.properties` is generated at server setup time and defines memory-related JVM settings based on the user-provided value for max heap size if aggressive memory tuning option was selected at setup. Most of the JVM options specified for both server and tools do not need customization after setup. The exception is the `-Xmx/Xms` options, which specify the maximum and initial JVM heap size. See the section on [Memory Allocation and Database Cache](#) for advice on tailoring the `-Xmx/Xms` values.

Other than altering the heap size of the server process (`start-ds`) or command-line tools, the most common change required to `java.properties` is when it is desired to update the JVM version. A single edit will apply the new JVM to all server and tool use.

Applying Changes Using `dsjavaproperties`

To apply the changes to the `config/java.properties` file, edit the file manually, and then run the `bin/dsjavaproperties` utility. The `dsjavaproperties` tool uses the information contained in the `config/java.properties` file to generate a `lib/set-java-home` script (or `lib\set-java-home.bat` on Microsoft Windows systems), which is used by the Directory Server and all of its supporting tools to identify the Java environment and its JVM settings. During the process, `dsjavaproperties` calculates an MD5 digest of the contents of the `config/java.properties` file and stores the digest in the generated `set-java-home` script.

The `dsjavaproperties` utility also performs some minimal validation whenever the property references a valid Java installation by verifying that `$(java-home)/bin/java` exists and is executable.

If you make any changes to the `config/java.properties` file but forget to run `bin/dsjavaproperties`, the Directory Server compares the MD5 digest with the version stored in `set-java-home` and sends a message to standard error if the digests differ:

```
WARNING -- File /ds/PingDirectory/config/java.properties has been edited without
running dsjavaproperties to apply the changes
```

To Update the Java Version in the Properties File

To change the version of java that is used by the server and tools, it is necessary to edit the `config/java.properties` file and apply the change by invoking `bin/dsjavaproperties` with no command line options. Also, the server must be restarted for the change to take affect.

Inside `config/java.properties`, alter the value of `default.java-home` to point to the java correct JRE. Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration.

```
$ bin/dsjavaproperties
```

To Regenerate the Java Properties File

The `dsjavaproperties` command provides a `--initialize` option that allows you to regenerate the Java Properties file specifically if you set up the Directory Server using standard memory usage but opt for aggressive memory tuning after setup. Rather than reconfigure the Java Properties file by re-running `setup` or manually editing the `java.properties` file, you can regenerate the properties file for aggressive memory tuning. Any existing file will be renamed with a ".old" suffix.

- Run the `dsjavaproperties` command to regenerate the java properties file for aggressive memory tuning:

```
$ bin/dsjavaproperties --initialize --jvmTuningParameter AGGRESSIVE
```

JVM Garbage Collection Using CMS

To ensure reliable server performance with Java, the Directory Server depends on Java's Concurrent Mark and Sweep process (CMS) for background garbage collection. There are several garbage collection options, with CMS being the ideal choice for consistent system availability. The CMS collector runs as one or more background threads, for the most part, within the JVM, freeing up space in JVM Heap from an area called the Old Generation. One of the criteria used by CMS to determine when to start background garbage collection is a parameter called `CMSInitiatingOccupancyFraction`. This percentage value, which applies to the Old Generation, is a recommendation for the JVM to initiate CMS when data occupancy in Old Generation reaches the threshold.

To understand this CMS property, it is important to know how large the Old Generation is and how much data in the Old Generation is expected to be occupied by the database cache. Ideally, the database cache takes less than 70% of the space available in the Old Generation, and the `CMSInitiatingOccupancyFraction` value of 80 leaves plenty of headroom to prevent the JVM from running out of space in Old Generation due to an inability for CMS to keep up. Because CMS takes processing resources away from the Directory Server, it is not recommended to set the `CMSInitiatingOccupancyFraction` at or below the expected database cache size, which would result in the constant running of CMS in the background. See the section on Memory Footprint and Database Cache for a description of determining Old Generation size.

When the CMS collection process cannot keep pace with memory demands in the Old Generation, the JVM will resort to pausing all application processing to allow a full garbage collection. This event, referred to as a *stop-the-world* pause, does not break existing TCP connections or alter the execution of the Directory Server requests. The goal in tuning CMS is to prevent the occurrence of these pauses. When one does occur, the Directory Server will generate an alert, after the pause, and record the pause time in the error log.

Because determining an ideal `CMSInitiatingOccupancyFraction` can be difficult, the approach we have taken is to warn if the Directory Server detects a garbage collection pause by generating a recommended value for the occupancy threshold based on the current amount of memory being consumed by the backend caches. Unfortunately, it is not possible for an administrator to determine the ideal occupancy threshold value in advance. Therefore, to warn of any impending garbage collection pauses, the Directory Server calculates a recommended value for the `CMSInitiatingOccupancyFraction` property and exposes it in the JVM Memory Usage monitor entry in the following attribute:

```
recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

Also, when you start the server, you will see an administrative alert indicating the current state of the `CMSInitiatingOccupancyFraction` and its recommended value.

```
$ bin/start-ds [20/April/2012:10:35:25 -0500] category=CORE severity=NOTICE
msgID=458886
msg="Ping Identity Directory Server 6.0.1.0 (build 20120418135933Z, R6226) starting up"
... (more output) ...

[20/April/2012:10:35:53 -0500] category=UBID_EXTENSIONS severity=NOTICE
msgID=1880555580 msg="Memory-intensive Directory Server
components are configured to consume 71750382 bytes of memory:
['userRoot local DB backend' currently consumes 26991632 bytes and
```

```
can grow to a maximum of 64323584 bytes, 'changelog cn=changelog backend'
currently consumes 232204 bytes and can grow to a maximum of 2426798 bytes,
'Replication Changelog Database' currently consumes 376661 bytes and can
grow to a maximum of 5000000 bytes). The configured value of
CMSInitiatingOccupancyFraction is 36 which is less than the minimum
recommended value (43) for the server's current configuration. Having
this value too low can cause the Concurrent Mark and Sweep garbage
collector to run too often, which can cause a degradation of throughput
and response time. Consider increasing the CMSInitiatingOccupancyFraction
value to at least the minimum value, preferably setting it to the
recommended value of 55 by editing the config/java.properties file,
running dsjavaproperties, and restarting the Directory Server.
If the server later detects that this setting actually leads to a
performance degradation, a separate warning message will be logged.
If this server has not yet been fully loaded with data, then you
can disregard this message"

[20/April/2012:10:35:53 -0500] category=CORE severity=NOTICE msgID=458887
msg="The Directory Server has started successfully"
```

The Directory Server only makes a recommendation if all of the backends are preloaded and the `CMSInitiatingOccupancyFraction` JVM property is explicitly set, which is done automatically. For example, if you installed the Directory Server and specified that the database be preloaded (or "primed") at startup, then the Directory Server can make a good recommendation for the Directory Server when a pause occurs. If the backend database cache is not full and has not been preloaded, then the recommended value may be an inaccurately low value.



Note: The generated value for the Directory Server property could change over time with each Directory Server build, Java release, or changes in data set. If the current value is fairly close to the recommended value, then there is no need to change the property unless the server experiences a JVM pause.

If the Directory Server experiences a JVM garbage collection pause, you can retrieve the recommended value from the server, reset the Directory Server property, run `dsjavaproperties`, and restart the server.

To Determine the `CMSInitiatingOccupancyFraction`

1. If you set the `Preload Database at startup` option during the installation, then skip to step 3. If you are not sure, retrieve the `prime-method` property for the backend as follows:

```
$ bin/dsconfig get-backend-prop --backend-name userRoot \
--property prime-method
```

2. If the `prime-method` property was not configured, use `bin/dsconfig` to set the property to `PRELOAD`, and then, restart the Directory Server to preload the database cache.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
--set prime-method:preload

$ bin/stop-ds
$ bin/start-ds
```

3. At startup, you will see an administrative message if the current `CMSInitiatingOccupancyFraction` property is below the recommended value. You can get

the recommended value from this message and change it in the `config/java.properties` file in step 5.

- If you were unable to see the recommended `CMSInitiatingOccupancyFraction` property at startup presented in the previous step, first you must pre-tune the value of the `CMSInitiatingOccupancyFraction` property to ensure that all of the data is imported into the server and preloading is enabled in the backend. Next, retrieve the recommended `CMSInitiatingOccupancyFraction` value by issuing the following search. If the `recommended-cms-initiating-occupancy-fraction-for-current-data-set` is not present, then make sure that the server has been restarted since enabling preload for the backend(s).

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectclass=ds-memory-usage-monitor-entry)" \
  cms-initiating-occupancy-fraction \
  recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

```
dn: cn=JVM Memory Usage,cn=monitor
cms-initiating-occupancy-fraction:80
recommended-cms-initiating-occupancy-fraction-for-current-data-set:55
```

- Open the `config/java.properties` file using a text editor, manually edit the `CMSInitiatingOccupancyFraction` or any other property to its recommended value in the `start-ds.java-args` property, and then, save the file when finished. (The following arguments are recommended for a Sun 5440 server. Contact your authorized support provider for specific assistance.):

```
start-ds.java-args=-d64 -server -Xmx20g -Xms20g -XX:MaxNewSize=1g
-XX:NewSize=1g -XX:ParallelGCThreads=16 -XX:+UseConcMarkSweepGC
-XX:+CMSConcurrentMTEnabled -XX:+CMSParallelRemarkEnabled
-XX:+CMSParallelSurvivorRemarkEnabled -XX:ParallelCMSThreads=8
-XX:CMSMaxAbortablePrecleanTime=3600000
-XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1
-XX:CMSInitiatingOccupancyFraction=55 -XX:+UseParNewGC
-XX:+UseBiasedLocking -XX:+UseLargePages
-XX:+HeapDumpOnOutOfMemoryError
```

The `-XX:ParallelGCThreads` should be limited to 16 (default) or to 8 for smaller systems. Also, the `-XX:ParallelCMSThreads` should be limited to 8.

- Run the `bin/dsjavaproperties` command to apply the changes.

```
$ bin/dsjavaproperties
```

- Restart the Directory Server.

Tuning For Disk-Bound Deployments

For best performance, configure the Directory Server to fully cache the DIT in the backend database cache. Directory Server configuration assumes this scenario. For databases too large to fit in memory, other options are available:

- Configure the server for a disk-bound data set (when the database is stored on an SSD, this configuration yields server performance that is comparable to a fully-cached scenario).
- Use uncached attributes and/or entries as described in the following section.

- Use a Directory Proxy Server in an entry-balancing deployment, which allows all data to be cached in a partitioned environment.

To Tune for Disk-Bound Deployments

To configure the server for a disk-bound configuration, follow these steps:

1. When installing the server, choose the "aggressive" option for JVM memory configuration and to preload the data when the server starts.
2. Set the `default-cache-mode` of the `userRoot` backend to `cache-keys-only`.
3. Set operating system `vm.swappiness` to 0 to protect the Directory Server JVM process from an overly aggressive filesystem cache.
4. When the data set is imported with the above settings, verify in the `import-ldif` output that the cached portions of the data set fit comfortably within the database cache.

Uncached Attributes and Entries

Although achieving optimal Directory Server performance requires that the entire data set be fully cached, there may be deployments in which fully caching the data set is not possible due to hardware or financial constraints, or in which acceptable performance can be achieved by only caching a portion of the data. The Directory Server already provides support for controlling caching on a per-database basis (e.g., to cache only certain indexes and/or system databases), but these features may not provide sufficient control over how memory is used, particularly with regard to which entries are included in the cache, and they do not provide any degree of control over caching only a portion of attributes.

To better address the needs of environments that require partial caching, the Directory Server provides two new options: the ability to exclude certain *entries* from the cache, and the ability to exclude certain *attributes* from the cache. The Directory Server uses an `uncached-id2entry` database container, which is similar to the `id2entry` database that maps an entry's unique identifier and its encoded representation. The `uncached-id2entry` database contains either complete and/or partial representations of entries that are intended to receive less memory for caching. For example, if an entry has a particularly large attribute and the system has hardware constraints on memory, then you can configure the system to not cache this particular attribute or entry. This functionality is only available for the local DB backend, which uses the Berkeley DB Java Edition database.

The `uncached-id2entry` database can be included in the set of databases to prime, but if priming is to be performed, it will only include internal nodes and not leaf nodes. For example, the internal nodes of the `uncached-id2entry` database will be included in the preload if the `prime-all-indexes` option is set to "true," or if the `system-index-to-prime-internal-nodes-only` option has a value of "uncached-id2entry".

Backup/Restore. There are no special considerations for backup and restore with regard to uncached entries and attributes. Backup will successfully save your database contents including uncached entries and attributes. Because of the way the server deals with changes to uncached

entry and uncached attribute configuration, there is no problem with restoring a backup that was taken with a different uncached entry configuration than is currently in place for the server. Any entries encoded in a manner that is inconsistent with the current uncached entry or uncached attribute configuration will be properly re-encoded whenever they are updated, or whenever the re-encode entries task is invoked.

Replication. Replication does not propagate information about which portions of entries may have been cached or uncached, nor does it require that different replicas have the same uncached attribute or uncached entry configuration.

LDIF Import/Export. When LDIF content is imported into the server, the uncached attribute and uncached entry configuration is used to determine on a per-entry basis whether some or all of the content for that entry should be written into the `uncached-id2entry` database. The determination is based on the current configuration and is completely independent of and unaware of the configuration that may have been in place when the LDIF data was initially exported. Neither the LDIF import nor export tools provide any options that specifically target only cached or only uncached content, but these tools do provide the ability to include or exclude entries using search filters, or to include or exclude specific attributes.

Server Access Log. Server access log messages may include `uncachedDataAccessed=true` in the result message for any operation in which it was necessary to access uncached data in the course of processing the associated request. For add, delete, modify, or modify DN result messages, `uncachedDataAccessed=true` indicates that at least a portion of the new or updated entry was written into the `uncached-id2entry` database, or that at least a portion of the updated entry was formerly contained in the `uncached-id2entry` database. For compare result messages, it indicates that at least a portion of the target entry was contained in the `uncached-id2entry` database and that data from the uncached portion of the entry was required to evaluate the assertion. For search result messages, it indicates that one or more of the entries evaluated as potential matches contained uncached data, and that data from the uncached portion of at least one entry was needed in determining what data should be returned to the client.

Uncached Entry/Attribute Properties. The Directory Server provides three new advanced properties on the Local DB Backend to control the caching mode for the `uncached-id2entry` database:

- **uncached-id2entry-cache-mode.** Specifies the cache mode that is used when accessing the records in the `uncached-id2entry` database. If the system has enough memory available to fully cache the internal nodes for this database, then `cache-keys-only` is recommended, otherwise it is better to select `no-caching` to minimize the amount of memory required for interacting with the `uncached-id2entry` database. For more information, see the *Ping Identity Directory Server Configuration Reference*.
- **uncached-attribute-criteria.** Specifies the criteria used to identify attributes that are written into the `uncached-id2entry` database, rather than the `id2entry` database. This property is only used for entries in which the associated `uncached-entry-criteria` does not indicate that the entire entry should be uncached. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all attributes are written into the `id2entry` database.

- **uncached-entry-criteria.** Specifies the criteria used to identify entries that are written into the `uncached-id2entry` database, rather than the `id2entry` database. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all entries are written into the `id2entry` database.

To Configure Uncached Attributes and Entries

The following procedure assumes that the `uncached-id2entry-cache-mode` property is set to the default value, `cache-keys-only`. For more information on the `uncached-id2entry` cache modes, see the *Ping Identity Directory Server Configuration Reference*.

1. Run `dsconfig` to uncache entries that match the criteria. Here, the filter will uncache all entries that have its location set to "austin" (i.e., `l=austin`).

```
$ bin/dsconfig create-uncached-entry-criteria \  
  --criteria-name "Fully Uncached l=austin" --type filter-based \  
  --set enabled:true --set "filter:(l=austin)"
```

2. Run `dsconfig` to uncache attributes that match the criteria (`attribute-type: jpegPhoto`). The `--type simple` option indicates that the simple uncached attribute criteria be used to specify the attribute-type that should be uncached, which in this example is `jpegPhoto`. For those entries that are fully stored in the `uncached-id2entry` database container, the uncached attribute will be ignored.

```
$ bin/dsconfig create-uncached-attribute-criteria \  
  --criteria-name "Uncached jpegPhoto" --type simple \  
  --set enabled:true --set attribute-type:jpegPhoto
```

3. Set the uncached properties for the `userRoot` backend.

```
$ bin/dsconfig set-backend-prop \  
  --backend-name userRoot \  
  --set "uncached-entry-criteria:Fully Uncached l=austin" \  
  --set "uncached-attribute-criteria:Uncached jpegPhoto"
```

4. Run the `re-encode-entries` tool to initiate a task that causes a local DB `userRoot` backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend. The following command initiates a task that re-encodes all fully-cached entries in the `userRoot` backend, rate-limited to no more than 100 entries per second.

```
$ bin/re-encode-entries --hostname directory.example.com --port 389 \  
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \  
  --backendID userRoot --skipFullyUncachedEntries \  
  --skipPartiallyUncachedEntries --ratePerSecond 100
```

Chapter

6

Configuring the Server

The out-of-the-box, initial configuration settings for the Ping Identity Directory Server provide an excellent starting point for most general-purpose Directory Server applications. However, additional tuning might be necessary to meet the performance, hardware, operating system, and memory requirements for your production environment.

The Directory Server stores its configuration settings in an LDIF file, `config/config.ldif`. Rather than editing the file directly, the Directory Server provides command-line and GUI tools that administrators can use to configure the server. The Directory Server also includes tools to create server groups, so that configuration changes can be applied to multiple servers at one time.

This chapter presents the following topics:

Topics:

- [Accessing the Directory Server Configuration](#)
- [About dsconfig Configuration Tool](#)
- [Using the Configuration API](#)
- [Configure the Server Using the Administrative Console](#)
- [Generating a Summary of Configuration Components](#)
- [About Root User, Administrator, and Global Administrators](#)
- [Managing Root Users Accounts](#)
- [Default Root Privileges](#)
- [Configuring Administrator Accounts](#)
- [Configuring a Global Administrator](#)
- [Configuring Server Groups](#)
- [Configuring Client Connection Policies](#)
- [Securing the Server with Lockdown Mode](#)
- [Configuring Maximum Shutdown Time](#)
- [Working with Referrals](#)
- [Configuring a Read-Only Server](#)
- [Configuring HTTP Access for the Directory Server](#)
- [Domain Name Service \(DNS\) Caching](#)
- [IP Address Reverse Name Lookups](#)
- [Configuring Traffic Through a Load Balancer](#)
- [Working with the Referential Integrity Plug-in](#)
- [Working with the Unique Attribute Plug-in](#)
- [Configuring Uniqueness Across Attribute Sets](#)
- [Working with the Last Access Time Plug-In](#)
- [Working with the Pass Through Authentication Plug-In](#)
- [Supporting Unindexed Search Requests](#)
- [Sun/Oracle Compatibility](#)

Accessing the Directory Server Configuration

The Ping Identity Directory Server configuration can be accessed and modified in the following ways:

- **Using the Administrative Console.** The Ping Identity Directory Server provides an Administrative Console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the `dsconfig` Command-Line Tool.** The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

About `dsconfig` Configuration Tool

The `dsconfig` tool is the text-based management tool used to configure the underlying Directory Server configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using `dsconfig` in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the Ping Identity Directory Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, `[389]` indicates that the default value for that field is port 389. You can press **Enter** to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

To Configure the Server Using dsconfig Interactive Mode

1. Launch the `dsconfig` tool in interactive command-line mode.

```
$ bin/dsconfig
```

2. Next, enter the LDAP connection parameters. Enter the Directory Server host name or IP address, or press **Enter** to accept the default.
3. Enter the number corresponding to the type of LDAP connection (1 for LDAP, 2 for SSL, 3 for StartTLS) that you are using on the Directory Server, or press **Enter** to accept the default (1).
4. Next, type the LDAP listener port number, or accept the default port. The default port is the port number of the server local to the tool.
5. Enter the user bind DN (default, `cn=Directory Manager`) and the bind DN password.
6. On the **Directory Server Configuration Console** main menu, type a number corresponding to the configuration that you want to change. Note that the number can change between releases or within the same release, depending on the options selected (for example, in cases where more expert level objects and properties are displayed).
In this example, select the number for Backend. Then, set the `db-cache-percent` to 40%. The optimal cache percentage depends on your system performance objectives and must be tuned as determined through analysis. In many cases, the default value chosen by the `setup` utility is sufficient.
7. On the **Backend management** menu, enter the number corresponding to view and edit an existing backend.
8. Select the backend to work with. In this example, using the basic object menu, only one backend that can be viewed in the directory, `userRoot`. Press **Enter** to accept the default.
9. From the **Local DB Backend** properties menu, type the number corresponding to the `db-cache-percent` property.
10. Enter the option to change the value, and then type the value for the `db-cache-percent` property. In this example, type 40 for "40 %".
11. Review the changes, and then type `f` to apply them.
Before you apply the change, the `dsconfig` interactive command-line mode provides an option to view the equivalent non-interactive command based on your menu selections. This is useful in building `dsconfig` script files for configuring servers in non-interactive or batch mode. If you want to view the equivalent `dsconfig` non-interactive command, type `d`. For more information, see [Getting the Equivalent dsconfig Non-Interactive Mode Command](#).
12. In the **Backend management** menu, type `q` to quit the `dsconfig` tool.

To View dsconfig Advanced Properties

For most configuration settings, some properties are more likely to be modified than others. The `dsconfig` interactive mode provides an option that hides or shows additional advanced properties that administrators might want to configure.

1. Repeat steps 1–9 in the previous section using `dsconfig` in Interactive Command-Line Mode.
2. From the **Local DB Backend properties** menu, type `a` to display the advanced properties, which toggles any hidden properties.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the Ping Identity Directory Server provides four different object menus that hide or expose configuration objects to the user. The purpose of object levels is to simply present only those properties that an administrator will likely use. The Object type is a convenience feature designed to unclutter menu readability.

The following object menus are available:

- **Basic.** Only includes the components that are expected to be configured most frequently.
- **Standard.** Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.
- **Advanced.** Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.
- **Expert.** Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

To Change the dsconfig Object Menu

1. Repeat steps 1–6 in the section using `dsconfig` in To Install the Directory Server in Interactive Mode.
2. On the **Ping Identity Directory Server configuration** main menu, type `o` (letter “o”) to change the object level. By default, Basic objects are displayed.
3. Enter a number corresponding to a object level of your choice: 1 for Basic, 2 for Standard, 3 for Advanced, 4 for Expert.
4. View the menu at the new object level. Additional configuration options for the Directory Server components are displayed.

Using dsconfig Interactive: Viewing Administrative Alerts

The `dsconfig` tool and the Administrative Console provide a useful feature that displays notifications for certain operations that require further administrator action to complete the process. If you change a certain backend configuration property, the admin action will appear in two places during a `dsconfig` interactive session: when configuring the property and before you apply the change. For example, if you change the `db-directory` property on the `userRoot` backend (that is, specify the path to the filesystem path that holds the Oracle Berkeley DB Java Edition backend files), you will see an admin action reminder during one of the steps (shown below).

The admin action alert will also appear as a final confirmation step. The alert allows you to continue and apply the change or back out of the configuration if the resulting action cannot be conducted at the present time. For example, after you type "f" to apply the `db-directory` property change, the admin alert message appears:

```
Enter choice [b]: f
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

* db-directory: Modification requires that the Directory Server be stopped,
the database directory manually relocated, and then the Directory Server
restarted. While the Directory Server is stopped, the directory and files
pertaining to this backend in the old database directory must be manually
moved or copied to the new location.

Continue? Choose 'no' to return to the previous step (yes / no) [yes]:
```

Currently, only a small set of properties display an admin action alert appear in `dsconfig` interactive mode and the Administrative Console. For more information on the properties, see the *Ping Identity Directory Server Configuration Reference*.

Using dsconfig in Non-Interactive Mode

The `dsconfig` non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the `dsconfig` command in non-interactive mode, which is convenient scripting applications. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the `dsconfig` tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The `--no-prompt` argument indicates that you want to use non-interactive mode. The `{subcommand}` is used to indicate which general action to perform. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the Directory Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the `dsconfig` tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \  
--hostname server.example.com \  
--port 389 \  
--bindDN uid=admin,dc=example,dc=com \  
--bindPassword password
```

If your system uses SASL GSSAPI (Kerberos), you can invoke `dsconfig` as follows:

```
$ bin/dsconfig --no-prompt list-backends \  
--saslOption mech=GSSAPI \  
--saslOption authid=admin@example.com \  
--saslOption ticketcache=/tmp/krb5cc_1313 \  
--saslOption useticketcache=true
```

The `{subcommandArgs}` argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the `--advanced` command-line option.



Note: Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To Configure the Server Using `dsconfig` Non-Interactive Mode

- Use the `dsconfig` command in non-interactive mode to change the amount of memory used for caching database contents and to specify common parent DN's that should be compacted in the underlying database.

```
$ bin/dsconfig set-backend-prop \  
--backend-name userRoot \  
--set db-cache-percent:40 \  
--add "compact-common-parent-dn:ou=accts,dc=example,dc=com" \  
--add "compact-common-parent-dn:ou=subs,dc=example,dc=com"
```

To View a List of `dsconfig` Properties

1. Use the `dsconfig` command with the `list-properties` option to view the list of all `dsconfig` properties. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties
```

2. Use the `dsconfig` command with the `list-properties` option and the `--complexity <menu level>` to view objects at and below the menu object level. You can also add the `--includeDescription` argument that includes a synopsis and description of each property in the output. Remember to add the LDAP connection parameters.

```
$ bin/dsconfig list-properties --complexity advanced --includeDescription
```

3. If the server is offline, you can run the command with the `--offline` option. You do not need to enter the LDAP connection parameters.

```
$ bin/dsconfig list-properties --offline --complexity advanced --includeDescription
```

You can also view the `<server-root>/docs/config-properties.txt` that contains the property information provided with the server.

Getting the Equivalent `dsconfig` Non-Interactive Mode Command

While the `dsconfig` non-interactive command-line mode is convenient for scripting and automating processes, obtaining the correct arguments and properties for each configuration change can be quite time consuming.

To facilitate easy and quick configuration, you can use an option to display the equivalent non-interactive command using `dsconfig` interactive mode. The command displays the equivalent `dsconfig` command to recreate the configuration in a scripted configuration or to more quickly enter any pending changes on the command line for another server instance.



Note: There are two other ways to get the equivalent `dsconfig` command. One way is by looking at the `logs/config-audit.log`. It might be more convenient to set the Directory Server up the way you want and then get the `dsconfig` arguments from the log. Another way is by configuring an option using the Administrative Console. The console shows the equivalent `dsconfig` command prior to applying the change.

To Get the Equivalent `dsconfig` Non-Interactive Mode Command

1. Using `dsconfig` in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").
2. Enter `d` to view the equivalent non-interactive command.
3. View the equivalent command (seen below), and then press **Enter** to continue. For example, based on an example in the previous section, changes made to the `db-cache-percent` returns the following:

```
Command line to apply pending changes to this Local DB Backend:
dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
```

The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using `dsconfig` Batch Mode

The Ping Identity Directory Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment. The `--no-prompt` option is required with `dsconfig` in batch mode.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process without applying any changes to the Directory Server. The `dsconfig` command supports a `--batch-continue-on-error` option which instructs `dsconfig` to apply all changes and skip any errors.

You can view the `logs/config-audit.log` file to review the configuration changes made to the Directory Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (`#`) for comments. The batch file also supports a `"\"` line continuation character for long commands that require multiple lines.

The Directory Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to Ping Identity Directory Server machines.

To Configure the Directory Server in `dsconfig` Batch Mode

1. Create a text file that lists each `dsconfig` command with the complete set of properties that you want to apply to the Directory Server. The items in this file should be in the same format as those accepted by the `dsconfig` command. The batch file can have blank lines for spacing and lines starting with a pound sign (`#`) for comments. The batch file also supports a `"\"` line continuation character for long commands that require multiple lines.

```
# Set the DB cache percent to 40
set-backend-prop --backend-name userRoot --set db-cache-percent:40

# Disable prime-method to none
set-backend-prop --backend-name userRoot --reset prime-method
```

2. Use `dsconfig` with the `--batch-file` option to read and execute the commands.

```
$ bin/dsconfig --hostname host1 --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" \
--bindPassword secret --no-prompt \
--batch-file /path/to/config-batch.txt
```

```
Batch file 'config-batch.txt' contains 2 commands:
Executing: set-backend-prop --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
--bindPassword ***** --no-prompt --backend-name userRoot --set db-cache-percent:40

Executing: set-backend-prop --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
--bindPassword ***** --no-prompt --backend-name userRoot --reset prime-method
```

Using the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the `application/json` content type.

The server includes a servlet extension that provides read and write access to the server's configuration over HTTP. The extension is enabled by default for new installations, and can be enabled for existing deployments by simply adding the extension to one of the server's HTTP Connection Handlers, as follows:

```
$ bin/dsconfig set-connection-handler-prop \
--handler-name "HTTPS Connection Handler" \
```

```
--add http-servlet-extension:Configuration
```

The API is made available on the HTTPS Connection handler's host:port in the `/config` context. Due to the potentially sensitive nature of the server's configuration, the HTTPS Connection Handler should be used for hosting the Configuration extension.

Authentication and Authorization with the Configuration API

Clients must use HTTP Basic authentication to authenticate to the Configuration API. If the username value is not a DN, then it will be resolved to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper, or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. For example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Configuration \
  --set "identity-mapper:Alternative Identity Mapper"
```

To access configuration information, users must have the appropriate privileges:

- To access the `cn=config` backend, users must have the `bypass-acl` privilege or be allowed access to the configuration using an ACI.
- To read configuration information, users must have the `config-read` privilege.
- To update the configuration, users must have the `config-write` privilege.

Relationship Between the Configuration API and the dsconfig Tool

The Configuration API is designed to mirror the `dsconfig` tool, using the same names for properties and object types. Property names are presented as hyphen case in `dsconfig` and as camel-case attributes in the API. In API requests that specify property names, case is not important. Therefore, `baseDN` is the same as `baseDn`. Object types are represented in hyphen case. API paths mirror what is in `dsconfig`. For example, the `dsconfig list-connection-handlers` command is analogous to the API's `/config/connection-handlers` path. Object types that appear in the schema URNs adhere to a `type:subtype` syntax. For example, a Local DB Backend's schema URN is `urn:unboundid:schemas:configuration:2.0:backend:local-db`. Like the `dsconfig` tool, all configuration updates made through the API are recorded in `logs/config-audit.log`.

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the `attributes` query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter.

Operations supported by the API are those typically found in REST APIs:

HTTP Method	Description	Related dsconfig Example
GET	Lists the properties of an object when used with a path representing an object, such as <code>/config/global-configuration</code> or <code>/config/</code>	<code>get-backend-prop</code> , <code>list-backends</code> , <code>get-global-configuration-prop</code>

HTTP Method	Description	Related dsconfig Example
	backends/userRoot. Can also list objects when used with a path representing a parent relation, such as /config/backends.	
POST	Creates a new instance of an object when used with a relation parent path, such as /config/backends.	create-backend
PUT	Replaces the existing properties of an object. A PUT operation is similar to a PATCH operation, except that the PATCH identifies the difference between an existing target object and a supplied source object. Only those properties in the source object are modified in the target object. The target object is specified using a path, such as /config/backends/userRoot.	set-backend-prop, set-global-configuration-prop
PATCH	Updates the properties of an existing object when used with a path representing an object, such as /config/backends/userRoot.	set-backend-prop, set-global-configuration-prop
DELETE	Deletes an existing object when used with a path representing an object, such as /config/backends/userRoot.	delete-backend

The OPTIONS method can also be used to determine the operations permitted for a particular path.

Object names, such as `userRoot` in the Description column, must be URL-encoded for use in the path segment of a URL. For example, `%20` must be used in place of spaces, and `%25` is used in place of the percent (%) character. The URL for accessing the HTTP Connection Handler object is:

```
/config/connection-handlers/http%20connection%20handler
```

GET Example

The following is a sample GET request for information about the `userRoot` backend:

```
GET /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
```

The response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://localhost:5033/config/backends/userRoot"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
```

```

"dbCheckpointInterval": "20 mb",
"dbCheckpointHighPriority": "false",
"dbCheckpointWakeupInterval": "1 m",
"dbCleanOnExplicitGC": "false",
"dbCleanerMinUtilization": "75",
"dbCompactKeyPrefixes": "true",
"dbDirectory": "db",
"dbDirectoryPermissions": "700",
"dbEvictorCriticalPercentage": "0",
"dbEvictorLruOnly": "false",
"dbEvictorNodesPerScan": "10",
"dbFileCacheSize": "1000",
"dbImportCachePercent": "60",
"dbLogFileMax": "50 mb",
"dbLoggingFileHandlerOn": "true",
"dbLoggingLevel": "CONFIG",
"dbNumCleanerThreads": "0",
"dbNumLockTables": "0",
"dbRunCleaner": "true",
"dbTxnNoSync": "false",
"dbTxnWriteNoSync": "true",
"dbUseThreadLocalHandles": "true",
"deadlockRetryLimit": "10",
"defaultCacheMode": "cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"enabled": "false",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "false",
"id2childrenIndexEntryLimit": "66",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"jeProperty": [
  "je.cleaner.adjustUtilization=false",
  "je.nodeMaxEntries=32"
],
"numRecentChanges": "50000",
"offlineProcessDatabaseOpenTimeout": "1 h",
"primeAllIndexes": "true",
"primeMethod": [
  "none"
],
"primeThreadCount": "2",
"primeTimeLimit": "0 ms",
"processFiltersWithUndefinedAttributeTypes": "false",
"returnUnavailableForUntrustedIndex": "true",
"returnUnavailableWhenDisabled": "true",
"setDegradedAlertForUntrustedIndex": "true",
"setDegradedAlertWhenDisabled": "true",
"subtreeDeleteBatchSize": "5000",
"subtreeDeleteSizeLimit": "5000",
"uncachedId2entryCacheMode": "cache-keys-only",
"writabilityMode": "enabled"
}

```

GET List Example

The following is a sample GET request for all local backends:

```

GET /config/backends/
Host: example.com:5033
Accept: application/scim+json

```

The response (which has been shortened):

```
{
```

```

"schemas": [
  "urn:ietf:params:scim:api:messages:2.0:ListResponse"
],
"totalResults": 24,
"Resources": [
  {
    "schemas": [
      "urn:unboundid:schemas:configuration:2.0:backend:ldif"
    ],
    "id": "adminRoot",
    "meta": {
      "resourceType": "LDIF Backend",
      "location": "http://localhost:5033/config/backends/adminRoot"
    },
    "backendID": "adminRoot",
    "backupFilePermissions": "700",
    "baseDN": [
      "cn=admin data"
    ],
    "enabled": "true",
    "isPrivateBackend": "true",
    "javaClass": "com.unboundid.directory.server.backends.LDIFBackend",
    "ldifFile": "config/admin-backend.ldif",
    "returnUnavailableWhenDisabled": "true",
    "setDegradedAlertWhenDisabled": "false",
    "writabilityMode": "enabled"
  },
  {
    "schemas": [
      "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
    ],
    "id": "ads-truststore",
    "meta": {
      "resourceType": "Trust Store Backend",
      "location": "http://localhost:5033/config/backends/ads-truststore"
    },
    "backendID": "ads-truststore",
    "backupFilePermissions": "700",
    "baseDN": [
      "cn=ads-truststore"
    ],
    "enabled": "true",
    "javaClass": "com.unboundid.directory.server.backends.TrustStoreBackend",
    "returnUnavailableWhenDisabled": "true",
    "setDegradedAlertWhenDisabled": "true",
    "trustStoreFile": "config/server.keystore",
    "trustStorePin": "*****",
    "trustStoreType": "JKS",
    "writabilityMode": "enabled"
  },
  {
    "schemas": [
      "urn:unboundid:schemas:configuration:2.0:backend:alarm"
    ],
    "id": "alarms",
    "meta": {
      "resourceType": "Alarm Backend",
      "location": "http://localhost:5033/config/backends/alarms"
    },
    ...
  }
]

```

PATCH Example

Configuration can be modified using the HTTP PATCH method. The PATCH request body is a JSON object formatted according to the SCIM patch request. The Configuration API, supports a subset of possible values for the path attribute, used to indicate the configuration attribute to modify.

The configuration object's attributes can be modified in the following ways. These operations are analogous to the `dsconfig modify-[object]` options.

- An operation to set the single-valued `description` attribute to a new value:


```
{
  "op" : "replace",
  "path" : "description",
  "value" : "A new backend."
}
```

is analogous to:

```
$ dsconfig set-backend-prop
  --backend-name userRoot \
  --set "description:A new backend"
```

- An operation to add a new value to the multi-valued `jeProperty` attribute:

```
{
  "op" : "add",
  "path" : "jeProperty",
  "value" : "je.env.backgroundReadLimit=0"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --add je-property:je.env.backgroundReadLimit=0
```

- An operation to remove a value from a multi-valued property. In this case, `path` specifies a SCIM filter identifying the value to remove:

```
{
  "op" : "remove",
  "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.cleaner.adjustUtilization=false
```

- A second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify, and a SCIM filter whose attribute is value:

```
{
  "op" : "remove",
  "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --remove je-property:je.nodeMaxEntries=32
```

- An option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value:

```
{
  "op" : "remove",
  "path" : "id2childrenIndexEntryLimit"
}
```

is analogous to:

```
$ dsconfig set-backend-prop --backend-name userRoot \
  --reset id2childrenIndexEntryLimit
```

The following is the full example request. The API responds with the entire modified configuration object, which may include a SCIM extension attribute `urn:unboundid:schemas:configuration:messages` containing additional instructions:

```
PATCH /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json

{
  "schemas" : [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

Example response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot2",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot2"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointIntervalBytes": "20 mb",
  "dbCheckpointIntervalHighPriority": "false",
  "dbCheckpointIntervalWakeup": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
}
```

```

"defaultCacheMode": "cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"description": "123", "enabled": "false",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "false",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"jeProperty": [ "\"je.env.backgroundReadLimit=0\""
],
"numRecentChanges": "50000",
"offlineProcessDatabaseOpenTimeout": "1 h",
"primeAllIndexes": "true",
"primeMethod": [
  "none"
],
"primeThreadCount": "2",
"primeTimeLimit": "0 ms",
"processFiltersWithUndefinedAttributeTypes": "false",
"returnUnavailableForUntrustedIndex": "true",
"returnUnavailableWhenDisabled": "true",
"setDegradedAlertForUntrustedIndex": "true",
"setDegradedAlertWhenDisabled": "true",
"subtreeDeleteBatchSize": "5000",
"subtreeDeleteSizeLimit": "5000",
"uncachedId2entryCacheMode": "cache-keys-only",
"writabilityMode": "enabled",
"urn:unboundid:schemas:configuration:messages:2.0": {
  "requiredActions": [
    {
      "property": "jeProperty",
      "type": "componentRestart",
      "synopsis": "In order for this modification to take effect,
        the component must be restarted, either by disabling and
        re-enabling it, or by restarting the server"
    },
    {
      "property": "id2childrenIndexEntryLimit",
      "type": "other",
      "synopsis": "If this limit is increased, then the contents
        of the backend must be exported to LDIF and re-imported to
        allow the new limit to be used for any id2children keys
        that had already hit the previous limit."
    }
  ]
}
}
}

```

Configuration API Paths

The Configuration API is available under the `/config` path. A full listing of supported sub-paths is available by accessing the base `/config/ResourceTypes` endpoint:

```

GET /config/ResourceTypes
Host: example.com:5033
Accept: application/scim+json

```

Sample response (abbreviated):

```

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {

```

```

"schemas": [
  "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
],
"id": "dsee-compat-access-control-handler",
"name": "DSEE Compat Access Control Handler",
"description": "The DSEE Compat Access Control
  Handler provides an implementation that uses syntax
  compatible with the Sun Java System Directory Server
  Enterprise Edition access control handler.",
"endpoint": "/access-control-handler",
"meta": {
  "resourceType": "ResourceType",
  "location": "http://example.com:5033/config/ResourceTypes/dsee-compat-access-
control-handler"
}
},
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
  ],
  "id": "access-control-handler",
  "name": "Access Control Handler",
  "description": "Access Control Handlers manage the
    application-wide access control. The server's access
    control handler is defined through an extensible
    interface, so that alternate implementations can be created.
    Only one access control handler may be active in the server
    at any given time.",
  "endpoint": "/access-control-handler",
  "meta": {
    "resourceType": "ResourceType",
    "location": "http://example.com:5033/config/ResourceTypes/access-control-
handler"
  }
}
},
{
  ...

```

The response's endpoint elements enumerate all available sub-paths. The path `/config/access-control-handler` in the example can be used to get a list of existing access control handlers, and create new ones. A path containing an object name such as `/config/backends/{backendName}`, where `{backendName}` corresponds to an existing backend (such as `userRoot`) can be used to obtain an object's properties, update the properties, or delete the object.

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted nor created. These paths can be differentiated from others by their singular, rather than plural, relation name (for example `global-configuration`).

Sorting and Filtering Objects

The Configuration API supports SCIM parameters for filter, sorting, and pagination. Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients can also specify sort parameters, or paging parameters. Include or exclude attributes can be specified in both get and list operations.

GET Parameter	Description
filter	Values can be simple SCIM filters such as <code>id eq "userRoot"</code> or compound filters like <code>meta.resourceType eq "Local DB Backend" and baseDn co "dc=example,dc=com"</code> .
sortBy	Specifies a property value by which to sort.

GET Parameter	Description
sortOrder	Specifies either ascending or descending alphabetical order.
startIndex	1-based index of the first result to return.
count	Indicates the number of results per page.

Updating Properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH. With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
}
```

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointInterval": "20 mb",
  "dbCheckpointHighPriority": "false",
  "dbCheckpointWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
```

```

"defaultCacheMode":
"cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"description": "abc",
"enabled": "true",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior":
"acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "true",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"numRecentChanges": "50000", "offlineProcessDatabaseOpenTimeout": "1 h",
"primeAllIndexes": "true",
"primeMethod": [
  "none"
],
"primeThreadCount": "2",
"primeTimeLimit": "0 ms",
"processFiltersWithUndefinedAttributeTypes": "false",
"returnUnavailableForUntrustedIndex": "true",
"returnUnavailableWhenDisabled": "true",
"setDegradedAlertForUntrustedIndex": "true",
"setDegradedAlertWhenDisabled": "true",
"subtreeDeleteBatchSize": "5000",
"subtreeDeleteSizeLimit": "100000",
"uncachedId2entryCacheMode": "cache-keys-only",
"writabilityMode": "enabled"
}

```

Administrative Actions

Updating a property may require an administrative action before the change can take effect. If so, the server will return 200 Success, and any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

For example, changing the `jeProperty` of a backend will result in the following:

```

"urn:unboundid:schemas:configuration:messages:2.0": {
  "required-actions": [
    {
      "property": "baseContextPath",
      "type": "componentRestart",
      "synopsis": "In order for this modification to take effect, the component
        must be restarted, either by disabling and re-enabling it, or
        by restarting the server"
    },
    {
      "property": {
        "type": "other",
        "synopsis": "If this limit is increased, then the
          contents of the backend must be exported to LDIF
          and re-imported to allow the new limit to be used
          for any id2children keys that had already hit the
          previous limit."
      }
    }
  ]
}

```

Updating Servers and Server Groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group. When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified. For example:

```
http://localhost:8082/config/backends/userRoot?applyChangeTo=single-server
```

Configuration API Responses

Clients of the API should examine the HTTP response code in order to determine the success or failure of a request. The following are response codes and their meanings:

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the <code>urn:unboundid:schemas:configuration:messages:2.0</code> object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers may respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

An application that uses the Configuration API should limit dependencies on particular text appearing in error message content. These messages may change, and their presence may

depend on server configuration. Use the HTTP return code and the context of the request to create a client error message. The following is an example encoded error message:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "status": 404,
  "scimType": null,
  "detail": "The Local DB Index does not exist."
}
```

Configure the Server Using the Administrative Console

The Ping Identity Directory Server provides a Administrative Console for server configuration and monitoring that has the same functionality as that of the `dsconfig` command. When logging on to the Administrative Console, the console does not persistently store any credentials for authenticating to the Directory Server but uses the credentials provided by the user when logging in. When managing multiple directory server instances, the provided credentials must be valid for each instance.

To Log onto the Administrative Console

To log into the console, enter a fully qualified DN (for example, `cn=admin2,cn=Administrators,cn=Admin Data`). See [Configuring a Global Administrator](#) for instructions.

1. Start the Directory Server.

```
$ bin/start-ds
```

2. Open a browser to `http://server-name:389/console/login`.
3. Enter the root user DN and password, then click **Login**.



The screenshot shows the Ping Identity PingData Administrative Console login interface. At the top left is the Ping Identity logo. Below it, the title "PingData Administrative Console" is displayed. The login form consists of three fields: "Server" with the value "ldaps://snapshot.lab:1636", "Username" with the value "directory manager", and "Password" which is masked with dots. A blue "Sign In" button is positioned below the password field. At the bottom left, the copyright notice "© 2015-2016 UnboundID Corp." is visible, and at the bottom right, there is a help icon (a question mark in a circle).

The console does not persistently store any credentials for accessing the Directory Server. Instead, it uses the credentials provided by the user when logging into the console. When managing multiple directory server instances, the provided credentials must be valid at each instance.

To Configure the Server Using the Console

1. Log into the Administrative Console. Click **Configuration** to open the **Configuration** menu, and then click **Backends**.

PingIdentity, PingData Administrative Console

snapshot.unboundid.lab:3389 /

Configuration ▾

LDAP Schema
Status

HTTP Servlet Cross Origin Policies ?

Core

Client Connection Policies ?

Connection Handlers ?

Global Configuration ?

Locations ?

Plugin Root ?

Virtual Attributes ?

Work Queue ?

Backends, Indexing, and Caching

Backends ?

Authentication and Password Management

Certificate Mappers ?

Identity Mappers ?

Password Generators ?

Password Policies ?

Password Validators ?

Logging, Monitoring

Alarm Manager

Alert Handlers

Gauges

Gauge Data Sources

Log Publishers

Log Retention Policies

Log Rotation Policies

Security and Authn

Key Manager Providers

Trust Manager Providers

Connection and Op

Connection Criteria

Request Criteria

Result Criteria

Search Entry Criteria

Search Reference Crit

Replication

Replication Assurance

- At the top of the Administrative Console page, click **Show Advanced Properties**. Click **Backends**. For this example, click **userRoot**.

PingIdentity. PingData Administrative Console

snapshot.unboundid.lab:3389 / Configuration ▾ / Backends /

Edit Local DB Backend

The Local DB Backend uses the Berkeley DB Java Edition to store user-provided data in a local repository.

General Configuration

Backend ID *

userRoot ?

Description

Enabled *

Enabled ?

Base DN *

Enter a value to add

dc=example,dc=com

Writability Mode *

enabled



Set Degraded Alert When Disabled

Enabled ?

Return Unavailable When Disabled

Enabled ?

Notification Manager



Is Private Backend

Enabled ⚠ ?

3. Change or enter values in the `userRoot` configuration. For this example, change the `DB Cache Percent` value to 40, and then click **Save**.

Generating a Summary of Configuration Components

The Directory Server provides a `config-diff` tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

Run the `config-diff --help` option to view other available tool options.

To Generate a Summary of Configuration Components

- Run the `config-diff` tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

```
$ bin/config-diff
```

- The following example compares the current configuration of the local server to the baseline, pre-installation configuration, ignoring any changes that could be made by the installer, and writes the output to the `configuration-steps.dsconfig` file. This provides a script that can be used to configure a newly installed server identically to the local server:

```
$ bin/config-diff --sourceLocal \  
--sourceBaseline \  
--targetLocal \  
--exclude differs-after-install \  
--outputFile configuration-steps.dsconfig
```

About Root User, Administrator, and Global Administrators

The Directory Server provides three different classes of administrator accounts: root user, administrator, and global administrator. The root user is the LDAP-equivalent of a UNIX super-user account and inherits its privileges from the default root user privilege set (see [Default Root Privileges](#)). The root user "account" is an entry that is stored in the server's configuration under the `cn=Root` DNs, `cn=config` and bypasses access control evaluation, and can be created manually, or with the `dsconfig` tool. This account has full access to the entire set of data in the Directory Information Tree (DIT) as well as full access to the server configuration and its operations. One important difference between other vendors' servers and the Directory Server's implementation is that the root user's rights are granted through a set of privileges. This allows the Directory Server to have multiple root users on its system if desired; however, the normal practice is to set up administrator user entries. Also, by default, the Root User has no resource limits.

The administrator user can have a full set of root user privileges but often has a subset of these privileges to limit the accessible functions that can be performed. The administrators entries typically have limited access to the entire set of data in the directory information tree

(DIT), which is controlled by access control instructions. These entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology. In some cases, administrator user accounts may be unavailable when the server enters lockdown mode unless the administrator is given the lock-down mode privilege.

A global administrator is primarily responsible for managing configuration server groups and can be used to log in to the Administrative Console. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at one time. Global Administrator(s) entries are stored in the `cn=admin data` backend along with other admin backend data and are always replicated between servers in a replication topology. These users can be assigned privileges like other admin users but are typically used to manage the data under `cn=admin data` and `cn=config`.

Managing Root Users Accounts

The Ping Identity Directory Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root users can be created and updated with the `dsconfig` tool. Root user entries are stored in the server's configuration. The following is a sample command to create a new root user:

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
--set last-name:Smith \
--set first-name:Joanne \
--set user-id:jsmith \
--set 'email-address:jsmith@example.com' \
--set mobile-telephone-number:8889997777 \
--set home-telephone-number:5556667777 \
--set work-telephone-number:4445556666
```

To limit full access to all of the Directory Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The Ping Identity Directory Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments. The following set of root privileges are available to each root user DN:

Table 4: Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.

Privilege	Description
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Server provides other privileges that are not assigned to the root user DN by default but can be added using the `ldapmodify` tool (see [Modifying Individual Root User Privileges](#)) for more information.

Table 5: Other Available Privileges

Privilege	Description
bypass-pw-policy	Allows the associated user bypass password policy rules and restrictions.
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.

Privilege	Description
jmx-write	Allows the associated user to perform JMX write operations.
permit-externally-processed-authentication	Allows the associated user accept externally processed authentication.
permit-proxied-mschapv2-details	Allows the associated user to permit MS-CHAP V2 handshake protocol.
proxied-auth	Allows the associated user to accept proxied authorization.

Configuring Administrator Accounts

An administrator account is any account in the user backend that is assigned one or more privileges, or given access to read and write operations beyond that of a normal user entry. The privilege mechanism is the same as that used for Root DN accounts and allows individual privileges to be assigned to an administrator entry.

Typically, administrator user entries are controlled by access control evaluation to limit access to the entire set of data in the Directory Information Tree (DIT). Fine-grained read and write access can be granted using the access control definitions available via the `aci` attribute. Administrator entries reside in the backend configuration (for example, `uid=admin,dc=example,dc=com`) and are replicated between servers in a replication topology.

The following examples show how to configure administrator accounts. The first procedure shows how to set up a single, generic `uid=admin,dc=example,dc=com` account with limited privileges. Note that if you generated sample data at install, you can view an example `uid=admin` entry using `ldapsearch`. The second example shows a more realistic example, where the user is part of the Administrators group. Note that both examples are based on a simple DIT. Actual deployment cases depends on your schema.

To Set Up a Single Administrator Account

1. Create an LDIF file with an example Administrator entry.

```
dn: uid=admin,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: Admin
uid: admin
cn: Admin User
sn: User
userPassword: password
```

2. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --defaultAdd --filename admin.ldif
```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to give full access to the new administrator. The ACI grants full access to all user attributes, but not to operational attributes. If you want to grant access to operational attributes as well as user attributes, use `(targetattr = "*" | "+")` in the access control instruction.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "**")
    (version 3.0; acl "Grant full access for the admin user";
      allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

4. Then add the entry using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename admin.ldif
```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `uid=Admin` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=admin)"
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base "(objectclass=*)" "+"
```

6. Add specific privileges to the Admin account. In this example, add the `password-reset` privilege to the admin account from the command line. After typing the privileges, press **CTRL-D** to process the modify operation.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
```

```
Processing MODIFY request for uid=admin,dc=example,dc=com
MODIFY operation successful for DN uid=admin,dc=example,dc=com
```

7. Assign a password policy for the Admin account. For example, create an "Admin Password Policy", then add the password policy to the account.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Admin Password Policy" \
  --set "description:Password policy for administrators" \
  --set password-attribute:userpassword \
  --set "default-password-storage-scheme:Salted 256-bit SHA-2" \
  --set password-change-requires-current-password:true \
  --set force-change-on-reset:true \
  --set "max-password-age:25w 5d" \
  --set grace-login-count:3
  --no-prompt
```

8. Apply the password policy to the account. In this example, the password policy is being added from the command line. The following `ldapmodify` command should be executed with a bind DN that has sufficient rights, such as a Root DN.

```
$ bin/ldapmodify
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password Policies,cn=config
```


To Change the Administrator Password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, if a root user password must be changed, use the `ldappasswordmodify` tool.

1. Open a text editor and create a text file containing the new password. In this example, name the file `rootuser.txt`.

```
$ echo password > rootuser.txt
```

2. Use `ldappasswordmodify` to change the root user's password.

```
$ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager"\
--bindPassword secret --newPasswordFile rootuser.txt
```

3. Remove the text file.

```
$ rm rootuser.txt
```

To Set Up an Administrator Group

The following example shows how to set up a group of administrators that have access rights to the whole Directory Server. The example uses a static group using the `GroupOfUniqueNames` object class.

1. Create an LDIF file with an example Administrator group, and save it as `admin-group.ldif`.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Dir Admins,ou=Groups,dc=example,dc=com
objectClass: groupofuniqueNames
objectClass: top
uniqueMember: uid=user.0, ou=People, dc=example,dc=com
uniqueMember: uid=user.1, ou=People, dc=example,dc=com
cn: Dir Admins
ou: Groups
```

2. Then, add the entries using the `ldapmodify` tool.

```
$ bin/ldapmodify --defaultAdd --filename admin-group.ldif
```

3. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to provide full access to the Directory Server to the new administrator. Save the file as `admin-aci.ldif`.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")
(targetattr != "aci")
(version 3.0; acl "allow all Admin group";
allow(all) groupdn = "ldap:///cn=Dir Admins,ou=Groups,dc=example,dc=com";)
```

4. Then, add the ACI using the `ldapmodify` tool:

```
$ bin/ldapmodify --filename admin-aci.ldif
```

5. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `cn=Dir Admins` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(cn=Dir Admins)"
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope base \
  "(objectclass=*)" "+"
```

6. Add specific privileges to each Admin account using an LDIF file, saved as `admin-priv.ldif`. In this example, add the `password-reset` privilege to the `user.0` admin account from the command line. Add the privilege using the `ldapmodify` tool. Repeat the process for the other administrators configured in the Admin group.

```
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
$ bin/ldapmodify --filename admin-priv.ldif
```

```
Processing MODIFY request for uid=user.0,dc=example,dc=com
MODIFY operation successful for DN uid=user.0,dc=example,dc=com
```

7. Assign a password policy for the Admin account using an LDIF file, saved as `admin-pwd-policy.ldif`. For example, create an "Admin Password Policy", then add the password policy to the account. Apply the password policy to the account using the `ldapmodify` tool.

```
dn: uid=user.0,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password Policies,cn=config
$ bin/ldapmodify --filename admin-pwd-policy.ldif
```

Configuring a Global Administrator

A global administrator is responsible for managing configuration server groups or can be used to log in to the Administrative Console. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at a time.

Global Administrator(s) are stored in the `cn=admin` data backend. These entries along with other admin backend data are always replicated between servers in a replication topology. Global Administrators can be assigned privileges like other admin users but are typically used to manage the data under `cn=admin` data and `cn=config`. You can create new global administrators and remove existing global administrators using the `dsframework` tool. The global administrator entries are located in the `cn=Administrators,cn=admin` data branch. Once you have set up a global admin, you can use it to log on to the Administrative Console. For example, the Administrative Console accepts the root user DN (for example, `cn=Directory`

Manager), the global admin ID (for example, `admin2`), or the full global admin DN (for example, `cn=admin2,cn=Administrators,cn=Admin Data`).

By default, the `dsframework` tool connects to any registered server interface when querying for configuration and status-related information. To connect over SSL or StartTLS, the `dsframework set-server-properties` command can be used to set the `preferredSecurity` property (possible values: `none`, `ssl`, or `starttls`). This feature allows the administrator to have control over which security protocol that the tool should use for such queries. To view the server properties, use the `dsframework list-server-properties` command.

To Create a Global Administrator

1. Use `dsframework` to create a new global administrator.

```
$ bin/dsframework create-admin-user \  
--userID admin2 --set password:secret
```

2. To verify the creation of the new administrator, use the `list-admin-users` subcommand with `dsframework`.

```
$ bin/dsframework list-admin-users  
id: admin2  
id: admin
```

To Remove a Global Administrator

1. Use `dsframework` to delete an existing global administrator.

```
$ bin/dsframework delete-admin-user --userID admin2
```

2. To verify the deletion of the global administrator, use the `list-admin-users` option with `dsframework`.

```
$ bin/dsframework list-admin-users  
id: admin
```

Configuring Server Groups

The Ping Identity Directory Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using `dsconfig`, then you can apply the change to the other servers in the group using the `--applyChangeTo server-group` option of the `dsconfig` non-interactive command. If you want to apply the change to one server in the group, use the `--applyChangeTo single-server` option. When using `dsconfig` in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

About the Server Group Example

You can create an administrative server group using the `dsframework` tool. The general process is to create a group, register each server, add each server to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in Replication Configuration.

The following example procedure adds three Directory Server instances into the server group labelled "group-one". The commands are run on `server1.example.com`.

Server	Host Name	LDAP Port
instance 1	server1.example.com	1389
instance 2	server2.example.com	2389
instance 3	server3.example.com	3389

To Create a Server Group

1. Create a group called "group-one" using `dsframework`.

```
$ bin/dsframework create-group --groupName group-one \  
--description "Server Group One"
```

2. Register each directory server that you want to add to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the `dsreplication enable` command.

```
$ bin/dsframework register-server \  
--serverID server1.example.com:1389 \  
--set hostname:server1.example.com \  
--set ldapport:1389 --set ldapEnabled:true  
  
$ bin/dsframework register-server \  
--serverID server2.example.com:2389 \  
--set hostname:server2.example.com \  
--set ldapport:2389 --set ldapEnabled:true  
  
$ bin/dsframework register-server \  
--serverID server3.example.com:3389 \  
--set hostname:server3.example.com \  
--set ldapport:3389 --set ldapEnabled:true
```

3. Add each directory server to the group.

```
$ bin/dsframework add-to-group \  
--groupName group-one \  
--memberName server1.example.com:1389  
  
$ bin/dsframework add-to-group \  
--groupName group-one \  
--memberName server2.example.com:2389  
  
$ bin/dsframework add-to-group \  
--groupName group-one \  
--memberName server3.example.com:3389
```

4. Set a global configuration property using the `dsconfig` tool.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:group-one
```

5. Test the server group. In this example, enable the log publisher for each directory server in the group, server-group, by using the `--applyChangeTo server-group` option.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --applyChangeTo server-group
```

6. View the property on the first directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
-----:-----
enabled : true
```

7. Repeat the step 6 on the second and third directory server instance.
8. Test the server group by disabling the log publisher on the first directory server instance by using the `--applyChangeTo single-server`.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:disabled \
  --applyChangeTo single-server
```

9. View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
-----:-----
enabled : false
```

10. View the property on the second directory server instance. Repeat this step on the third directory server instance to verify that the property is still enabled on that server.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled
```

```
Property : Value(s)
-----:-----
enabled : true
```

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client

address, protocol, communication security, and authentication state and identity. The client connection policy, however, does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time, which is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria.** The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.
- **Evaluation order index.** If multiple client connection policies are defined in the server, then each of them must have a unique value for the **evaluation-order-index** property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.

- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using StartTLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

You can restrict the types of search filters that a given client may be allowed to use to prevent the use of potentially expensive filters, like range or substring searches. You can use the `allowed-filter-type` property to provide a list of filter types that may be included in the search requests from clients associated with the client connection policy. This setting will only be used if search is included in the set of allowed operation types. This restriction will only be applied to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

The `minimum-substring-length` property can be used to specify the minimum number of non-wildcard characters in a substring filter. Any attempt to use a substring search with an element containing fewer than this number of bytes will be rejected. For example, the server can be configured to reject filters like `"(cn=a*)"` and `"(cn=ab*)"`, but to allow `"(cn=abcde*)"`. This property setting will only be used if search is included in the set of allowed operation types and at least one of `sub-initial`, `sub-any`, or `sub-final` is included in the set of allowed filter types.

There are two primary benefits to enforcing a minimum substring length:

- Allowing very short substrings can require the server to perform more expensive processing. The search requires a lot more server effort to assemble a candidate entry list for short substrings because the server has to examine a lot more index keys.
- Allowing very short substrings makes it easier for a client to put together a series of requests to retrieve all the data from the server (a process known as "trawling"). If a malicious user wants to obtain all the data from the server, then it is easier to issue 26 requests

like "(cn=a*)", "(cn=b*)", "(cn=c*)", ..., "(cn=z*)" than if the user is required to do something like "(cn=aaaaa*)", "(cn=aaaab*)", "(cn=aaaac*)", ..., "(cn=zzzzz*)".

Setting Resource Limits

Your client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. You can limit the total number of connections to a server from a particular client or from clients that match specified criteria. You can also limit the duration of the connection.

A client connection policy may only be used to enforce additional restrictions on a client connection. You can never use it to grant a client capabilities that it would not otherwise have.

Any change to any of these new configuration properties will only impact client connections that are assigned to the client connection policy after the change is made. Any connection associated with the client connection policy before the configuration change was made will continue to be subject to the configuration that was in place at the time it was associated with that policy.

Table 6: Resource Limiting Properties

Property	Description
maximum-concurrent-connections	<p>Specifies the maximum number of client connections that can be associated with that client connection policy at any given time. The default value of zero indicates that no limit will be enforced.</p> <p>If the server already has the maximum number of connections associated with a client connection policy, then any attempt to associate another connection with that policy (e.g., newly-established connections or an existing connection that has done something to change its client connection policy, such as perform a bind or StartTLS operation) will cause that connection to be terminated.</p>
terminate-connection	<p>Specifies that any client connection for which the client connection policy is selected (whether it is a new connection or an existing connection that is assigned to the client connection policy after performing a bind or StartTLS operation) will be immediately terminated.</p> <p>This property can be used to define criteria for connections that you do not want to be allowed to communicate with the Directory Server.</p>
maximum-connection-duration	<p>Specifies the maximum length of time that a connection associated with the client connection policy can remain established to the Directory Server, regardless of the amount of activity on that connection.</p> <p>A value of "0 seconds" (default) indicates that no limit will be enforced. If a connection associated with the client connection policy has been established for longer than this time, then it will be terminated.</p>
maximum-idle-connection-duration	<p>Specifies the maximum length of time that a connection associated with the client connection policy can remain established with the Directory Server without any requests in progress.</p> <p>A value of "0 seconds" (default) indicates that no additional limit will be enforced on top of whatever idle time limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective idle time limit for any client connection will be the smaller of the maximum-</p>

Property	Description
	<p><code>idle-connection-duration</code> from the client connection policy and the idle time limit that would otherwise be in effect for that client.</p> <p>This property can be used to apply a further restriction on top of any value that may be enforced by the <code>idle-time-limit</code> global configuration property (which defines a default idle time limit for client connections) or the <code>ds-rlim-idle-time-limit</code> operational attribute (which may be included in a user entry to override the default idle time limit for that user).</p>
<code>maximum-operation-count-per-connection</code>	<p>Specifies the maximum number of operations that a client associated with the client connection policy will be allowed to request. A value of zero (default) indicates that no limit will be enforced. If a client attempts to request more than this number of operations on the same connection, then that connection will be terminated.</p>
<code>maximum-concurrent-operations-per-connection</code>	<p>Specifies the maximum number of operations that may be active at any time from the same client. This limit is only applicable to clients that use asynchronous operations with multiple outstanding requests at any given time.</p> <p>A value of zero (default) indicates that no limit will be enforced.</p> <p>If a client already has the maximum number of outstanding requests in progress and issues a new request, then that request will be delayed and/or rejected based on the value of the <code>maximum-concurrent-operation-wait-time-before-rejecting</code> property.</p>
<code>maximum-concurrent-operation-wait-time-before-rejecting</code>	<p>Specifies the maximum length of time that a client connection should allow an outstanding operation to complete if the maximum number of concurrent operations for a connection are already in progress when a new request is received on that connection.</p> <p>A value of "0 seconds" (default) indicates that any new requests received while the maximum number of outstanding requests are already in progress for that connection will be immediately rejected.</p> <p>If an outstanding operation completes before this time expires, then the server may be allowed to process that operation. If the time expires, the new request will be rejected.</p>
<code>allowed-request-control</code>	<p>Specifies the OIDs of the request controls that clients associated with the client connection policy will be allowed to use.</p> <p>If any <code>allowed-request-control</code> OIDs are specified, then any request which includes a control not in that set will be rejected. If no <code>allowed-request-control</code> values are specified (default), then any control whose OID is not included in the set of <code>denied-request-control</code> values will be allowed.</p>
<code>denied-request-control</code>	<p>Specifies the OIDs of the request controls that clients associated with the client connection policy will not be allowed to use. If there are any <code>denied-request-control</code> values, then any request containing a control whose OID is included in that set will be rejected.</p> <p>If there are no <code>denied-request-control</code> values (default), then any request control will be allowed if the <code>allowed-request-control</code> property is also empty, or only those controls whose OIDs are included in the set of <code>allowed-request-control</code> values will be allowed if at least one <code>allowed-request-control</code> value is provided.</p>
<code>allowed-filter-type</code>	<p>Specifies the types of components which may be used in filters included in search operations with a non-base scope that are requested by clients associated with the client connection policy. Any non-base scoped search</p>

Property	Description
	<p>request whose filter contains a component not included in this set will be rejected. The set of possible filter types include:</p> <ul style="list-style-type: none"> > and > or > not > equality > sub-initial > sub-any > sub-final > greater-or-equal > less-or-equal > approximate-match > extensible-match <p>By default, all filter types will be allowed. Also note that no restriction will be placed on the types of filters which may be used in searches with a base scope.</p>
allow-unindexed-searches	<p>Specifies whether clients associated with the client connection policy will be allowed to request searches which cannot be efficiently processed using the configured set of indexes. Note that clients will still be required to have the <code>unindexed-search</code> privilege, so this option will not grant the ability to perform unindexed searches to clients that would not have otherwise had that ability, but it may be used to prevent clients associated with the client connection policy from requesting unindexed searches when they might have otherwise been allowed to do so.</p> <p>By default, this has a value of "true", indicating that any client associated with the client connection policy that has the <code>unindexed-search</code> privilege will be allowed to request unindexed searches.</p>
minimum-substring-length	<p>Specifies the minimum number of bytes, which may be present in any sub-Initial, subAny, or subFinal element of a substring search filter component in a search with a non-baseObject scope. A value of one (which is the default) indicates that no limit will be enforced. This property may be used to prevent clients from issuing overly-vague substring searches that may require the Installing the Directory Server to examine too many entries over the course of processing the request.</p>
maximum-search-size-limit	<p>Specifies the maximum number of entries that may be returned from any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit and will never increase any limit that may already be in effect for the client via the <code>size-limit</code> global configuration property or the <code>ds-rlim-size-limit</code> operational attribute.</p> <p>A value of zero (default) indicates that no additional limit will be enforced on top of whatever size limit might already be in effect for an associated connection.</p> <p>If a nonzero value is provided, then the effective maximum size limit for any search operation requested by the client will be the smaller of the size limit from that search request, the <code>maximum-search-size-limit</code> from the client connection policy, and the size limit that would otherwise be in effect for that client.</p>

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of k (to indicate that the integer should be interpreted as thousands), m (to indicate that the integer should be interpreted as millions), or g (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

- > 1/s (no more than one operation over a one-second interval)
- > 10K/5h (no more than ten thousand operations over a five-hour interval)
- > 5m/2d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

- > Two external LDAP clients are allowed to bind to the Directory Server.
- > Client 1 should be allowed to open only 1 connection to the server.
- > Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

- **Client 1 Connection Policy**, which only allows client 1, with an evaluation order index of 1.
- **Client 2 Connection Policy**, which only allows client 2, with an evaluation order index of 2.
- **Unauthenticated Connection Policy**, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

- The `user-auth-type` must not include none, so that it will only apply to authenticated client connections.
- The `included-user-base-dn` should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation order index. Policies with a lower evaluation order index value will be examined before those with a higher evaluation order index value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest evaluation order index value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be re-evaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthenticated Connection Policy, because it is the client

connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthenticated Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthenticated Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using the Console

1. Open the Administrative Console. Provide a username and password, and then click **Login**.
2. In the **Core Server** section, click **Client Connection Policies**. If you do not see **Client Connection Policies** on the menu, change the Object Types filter to **Standard**.
3. Click **Add New** to add a new policy.
4. Enter a Policy ID. If you want to base your new client connection policy on an existing policy, select it from the **Template** menu.
5. Configure the properties of the client connection policy. To enable the policy, select **Enabled**.
6. Enter the order in which you want the new policy to be evaluated in the **Evaluation Order Index** box, and then click **Continue**. A policy with a lower index is evaluated before a policy with a higher index. The Directory Server uses the first evaluated policy that applies to a client connection.
7. Select the connection criteria that match the client connection for this policy. Click **View and edit** to change the criteria. Click **Select New** to add new criteria. Select the operations allowed for clients that are members of this connection group. Use the **Add and Remove** buttons to make operations available to clients. Specify the extended operations that clients are allowed and denied to use.
8. Enter the type of authorization allowed and the SASL mechanisms that are allowed and denied in response to client requests.

9. Check the **Include Backend Subtree Views** check box if you want to automatically include the subtree views of backends configured in the Directory Server. You can also choose to include and exclude specific base DNs using the appropriate fields.
10. Once you have finished configuring the properties of your client connection policy, click **Confirm then Save** to review the `dsconfig` command equivalent and save your changes. Click **Save Now** to save your changes without first reviewing the `dsconfig` output.

To Configure a Client Connection Policy Using `dsconfig`

You can configure a client connection policy using the `dsconfig` tool in interactive mode from the command line. You can access the **Client Connection Policy** menu on the **Standard objects** menu.

1. Use the `dsconfig` tool to create and configure a client connection policy. Specify the host name, connection method, port number, and bind DN as described in previous procedures.
2. On the Directory Server main menu, change to the **Standard objects** menu by entering `o` and then entering the number for the Standard menu.
3. On the Directory Server main menu, enter the number associated with **Client Connection Policy**.
4. On the **Client Connection Policy management** menu, type the number corresponding to **Create a new connection policy**.
5. Enter `n` to create a new client connection policy from scratch.
6. Next, enter a name for the new client connection policy.
7. On the **Enabled Property** menu, select `true` to enable the connection policy.
8. On the **Evaluation-Order Property** menu, type a value between 0 and 2147483647 to set the evaluation order for the policy. A client connection policy with a lower evaluation-order will be evaluated before one with a higher number. For this example, type `9999`.
9. On the **Client Connection Policy** menu, review the configuration. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Restricting Server Access Based on Client IP Address

Another common use case is to limit client access to the Directory Server. Two methods are available:

- **Connection Handlers.** You can limit the IP addresses using the LDAP or LDAPS connection handlers. The connection handlers provide an `allowed-client` property and a `denied-client` property. The `allowed-client` property specifies the set of allowable

address masks that can establish connections to the handler. The `denied-client` property specifies the set of address masks that are not allowed to establish connections to the handler.

- **Client Connection Policies.** You can take a more fine-grained approach by restricting access by configuring a new Client Connection Policy, then create a new connection criteria and associate it with the connection policy. Connection criteria define sets of criteria for grouping and describing client connections based on a number of properties, including the protocol, client address, connection security, and authentication state for the connection. Each client connection policy may be associated with zero or more Connection Criteria, and server components may use Connection Criteria to indicate which connections should be processed and what kind of processing should be performed (e.g., to select connections and/or operations for filtered logging, or to classify connections for network groups).

To Restrict Server Access Using the Connection Handlers

- Use `dsconfig` to set the `allowed-client` property for the LDAP connection handler. You should specify the address mask for the range of allowable IP addresses that can establish connections to the Directory Server. You should also specify the loopback address, `127.0.0.1`, so that you will still be able to configure the server using the `dsconfig` tool on the local host.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set "allowed-client:10.6.1.*" \
  --set allowed-client:127.0.0.1
```

To Restrict Server Access Using Client Connection Policies

1. Create a simple connection criteria. The following example uses the `dsconfig` tool in non-interactive mode. It allows only the Directory Server's IP address and loopback to have access.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-adrs \
  --add included-client-address:10.6.1.80 \
  --add included-client-address:127.0.0.1
```

2. Assign the criteria to the client connection policy. After you have run the following command, access is denied to remote IP addresses. The Directory Server does not require a restart.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name new-policy \
  --set connection-criteria:allowed-ip-adrs
```

3. Add a remote IP range to the criteria. For this example, add `10.6.1.*`. Access from any remote servers is allowed. The Directory Server does not require a restart.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-adrs \
  --add "included-client-address:10.6.1.*"
```

4. To restore default behavior, you can remove the criteria from the connection policy. The Directory Server does not require a restart. Remember to include the LDAP or LDAPS connection parameters (e.g., `hostname`, `port`, `bindDN`, `bindPassword`) with the `dsconfig` command.

```
$ bin/dsconfig set-client-connection-policy-prop \
```

```
--policy-name new-policy --remove connection-criteria:allowed-ip-addr
```

To Automatically Authenticate Clients that Have a Secure Communication Channel

The Directory Server provides an option to automatically authenticate clients that have a secure communication channel (either SSL or StartTLS) and presented their own certificate. This option is disabled by default, but when enabled, the net effect will be as if the client issued a SASL EXTERNAL bind request on that connection.

This option will be ignored if the client connection is already authenticated (e.g., because it is using StartTLS but the client had already performed a bind before the StartTLS request). If the bind attempt fails, then the connection will remain unauthenticated but usable. If the client subsequently sends a bind request on the connection, then it will be processed as normal and any automatic authentication will be destroyed.

- Run the following `dsconfig` command.

```
$ bin/dsconfig set-connection-handler-prop \  
--handler-name "LDAPS Connection Handler" \  
--set "auto-authenticate-using-client-certificate:true"
```

Securing the Server with Lockdown Mode

The Directory Server provides tools to enter and leave lockdown mode if the server requires a security lockdown. In lockdown mode, only users with the `lockdown-mode` privilege can perform operations, while those without the privilege are rejected. Root users have this privilege by default; other administrators can be given this privilege.

The Directory Server can be manually placed into lockdown mode to perform some administrative operation while ensuring that other client requests are not allowed to access any data in the server. In addition, some configuration problems (particularly problems that could lead to inadvertent exposure of sensitive information, like an access control rule that cannot be properly parsed) cause the server to place itself in lockdown mode, so that an administrator can manually correct the problem. Lockdown mode does not persist across restarts. The directory server can be taken out of lockdown mode by using either the `leave-lockdown-mode` tool or by restarting the server. If administrators want to start a server in lockdown mode, they can use the `start-ds --lockdownMode` option.

Any client request to the Directory Server in lockdown mode receives an "Unavailable" response.

To Manually Enter Lockdown Mode

- Use `enter-lockdown-mode` to begin a lockdown mode.

```
$ bin/enter-lockdown-mode
```


To Leave Lockdown Mode

- Use `leave-lockdown-mode` to end lockdown mode.

```
$ bin/leave-lockdown-mode
```

To Start a Server in Lockdown Mode

- Use the `--lockdownMode` option with the `start-ds` tool to start a server in lockdown mode.

```
$ bin/start-ds --lockdownMode
```

Configuring Maximum Shutdown Time

During shutdown, some database checkpointing and cleaning threads may remain active even after the default time period on systems with very large or very busy database backends. If checkpointing or cleaning is aborted prematurely, it could possibly lead to significantly longer startup times for the Directory Server. The Directory Server provides an option for administrators to set the maximum time a shutdown process should take. When a shutdown process is initiated, the server begins stopping all of its internal components and waits up to 5 minutes for all threads to complete before exiting.

Administrators can use the `dsconfig` tool to increase the maximum shutdown time to allow database operations to complete.

To Configure the Maximum Shutdown Time

- Use the `dsconfig` tool to increase the maximum shutdown time for your system. The following command increases the maximum shutdown time from 5 minutes to 6 minutes. The command allows time values of w (weeks), d (days), h (hours), m (minutes), s (seconds), ms (milliseconds).

```
$ bin/dsconfig set-global-configuration-prop --set "maximum-shutdown-time:6 m"
```

Remember to include the LDAP or LDAPS connection parameters (e.g., host name, port, `bindDN` and `bindDN` password) with the `dsconfig` command.

The `maximum-shutdown-time` property can also be changed using the `dsconfig` tool in interactive mode. From the main menu, select **Global Configuration**, and then select the option to display advanced properties.

Working with Referrals

A referral is a redirection mechanism that tells client applications that a requested entry or set of entries is not present on the Directory Server but can be accessed on another server. Referrals

can be used when entries are located on another server. The Directory Server implements two types of referrals depending on the requirement.

- **Referral on Update Plug-in.** The Directory Server provides a Referral on Update Plug-in to create any referrals for update requests (add, delete, modify, or modDN operations) on read-only servers. For example, given two replicated directory server where one server is a master (read-write) and the other, a read-only server, you can configure a referral for any client update requests on the second directory server to point to the master server. If a client application sends an add request, for example, on the second directory server, the directory server responds with a referral that indicates any updates should be made on the master server. All read requests on the read-only server will be processed as normal.
- **Smart Referrals.** The Directory Server supports smart referrals that map an entry or a specific branch of a DIT to an LDAP URL. Any client requests (reads or writes) targeting at or below the branch of the DIT will send a referral to the server designated in the LDAP URL.

Specifying LDAP URLs

Referrals use LDAP URLs to redirect a client application's request to another server. LDAP URLs have a specific format, described in RFC 4516 and require that all special characters be properly escaped and any spaces indicated as "%20". LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base-dn?attributes?scope?filter
```

where

- *ldap[s]* indicates the type of LDAP connection to the Directory Server. If the Directory Server connects over a standard, non-encrypted connection, then *ldap* is used; if it connects over SSL, then *ldaps* is used. Note that any search request initiated by means of an LDAP URL is anonymous by default, unless an LDAP client provides authentication.
- *hostname* specifies the host name or IP address of the Directory Server.
- *port* specifies the port number of the Directory Server. If no port number is provided, the default LDAP port (389) or LDAPS port (636) is used.
- *base-dn* specifies the distinguished name (DN) of an entry in the DIT. The Directory Server uses the base DN as the starting point entry for its searches. If no base DN is provided, the search begins at the root of the DIT.
- *attributes* specifies those attributes for which the Directory Server should search and return. You can indicate more than one attribute by providing a comma-separated list of attributes. If no attributes are provided, the search returns all attributes.
- *scope* specifies the scope of the search, which could be one of the following: *base* (only search the specified base DN entry), *one* (only search one level below the specified base DN), *sub* (search the base entry and all entries below the specified base DN). If no scope is provided, the server performs a base search.
- *filter* specifies the search filter to apply to entries within the scope of the search. If no filter is provided, the server uses +.

Creating Referrals

You can create a smart referral by adding an entry with the referral and `extensibleObject` object classes or adding the object classes to a specific entry. The referral object class designates the entry as a referral object. The `extensibleObject` object class allows you to match the target entry by matching any schema attribute. The following example shows how to set up a smart referral if a portion of a DIT is located on another server.

To Create a Referral

1. Create an LDIF file with an entry that contains the referral and `extensibleObject` object classes.

```
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
objectClass: top
objectClass: referral
objectClass: extensibleObject
ou: Engineering Team1
ref: ldap://server2.example.com:6389/ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

2. On the first server, add the referral entry using the `ldapmodify` command.

```
$ bin/ldapmodify --defaultAdd --fileName referral-entry.ldif
```

3. Verify the addition by searching for a user.

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com "(uid=user.4)"
```

```
SearchReference(referralURLs={ldap://server2.example.com:6389/
ou=EngineeringTeam1,ou=People,dc=example,dc=com??sub?})
```

To Modify a Referral

- Use `ldapmodify` with the `manageDSAIT` control to modify the `ref` attribute on the referral entry.

```
$ bin/ldapmodify --control manageDSAIT
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
changetype: modify
replace: ref
ref: ldap://server3.example.com/ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

To Delete a Referral

- Use `ldapdelete` with the `manageDSAIT` control to delete the referral entry.

```
$ bin/ldapdelete \
--control manageDSAIT "ou=EngineeringTeam1,ou=People,dc=example,dc=com"
```

Configuring a Read-Only Server

The Ping Identity Directory Server provides a means to configure a hub-like, read-only directory server for legacy systems that require it. The read-only directory server participates in replication but cannot respond to any update requests from an external client. You can configure the Directory Server by setting the writability mode to internal-only, which makes the server operate in read-only mode. Read-only mode directory servers can process update operations from internal operations but reject any write requests from external clients. Because the Directory Server cannot accept write requests, you can configure the server to send a referral, which redirects a client's request to a master server. The client must perform the operation again on the server named in the referral.



Note: For Implementers of Third Party Extensions. Many Server SDK extensions use the `InternalConnection` interface to process operations in the server, rather than issuing LDAP requests over the network. If an extension does so in response to an external update request, then any Directory Server using that extension will effectively respond to external update requests, even though the Directory Server is configured to operate in read-only mode, as described above. One possible workaround is to split the extension into two extensions, one for reads and one for writes, then disabling (or not deploying) the write-only extension when configuring a Directory Server in read-only mode.

To Configure a Read-Only Server

1. Install two replicating directory servers. See [Replication Configuration](#) for various ways to set up your servers.
2. On the second server, use the `dsconfig` command to set the writability mode of the server to internal-only.

```
$ bin/dsconfig set-global-configuration-prop \  
  --set writability-mode:internal-only
```

3. On the second server, use the `dsconfig` tool to create a referral that instructs the server to redirect client write requests under `dc=example,dc=com` to `server1.example.com:1389`. The referral itself is defined as a plugin of type `Referral` on `Update`. This command sets up the server to process read operations but redirects all write operations under `dc=example,dc=com` to another server.

```
$ bin/dsconfig create-plugin --plugin-name "Refer Updates" \  
  --type referral-on-update \  
  --set enabled:true \  
  --set referral-base-url:ldap://server1.example.com:1389/ \  
  --set "base-dn:dc=example,dc=com"
```

- To test the referral, attempt to modify an entry and confirm that the server responds with the result code of 10. The resulting message is available in the server's access log.

```
$ bin/ldapmodify -p 2389 -D "cn=Directory Manager" -w password
dn: uid=user.12,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 408 555 1155
```

```
[06/Aug/2012:15:28:21.468 -0400] MODIFY
RESULT conn=86 op=1 msgID=1 requesterIP="127.0.0.1"
dn="uid=user.12,ou=People,dc=example,dc=com" resultCode=10
referralURLs="ldap://server1.example.com:1389/uid=user.12,
ou=People,dc=example,dc=com" etime=0.223
```

Configuring HTTP Access for the Directory Server

Although most clients communicate with the Ping Identity Directory Server using LDAP, the server also provides support for an HTTP connection handler that uses Java servlets to serve content to clients over HTTP. PingData offers an extension that uses this HTTP connection handler to add support for the System for Cross-domain Identity Management (SCIM) protocol. Third-party developers can also use the Server SDK to write extensions that leverage this HTTP support.

The following sections describe how to configure HTTP servlet extensions and how to configure an HTTP connection handler.

Configuring HTTP Servlet Extensions

To use the HTTP connection handler, you must first configure one or more servlet extensions. Servlet extensions are responsible for obtaining Java servlets (using the Java Servlet 3.0 specification as described in JSR 315) and registering them to be invoked using one or more context paths. If you plan to deploy the SCIM extension, then you should follow the instructions in Chapter 24, "*Managing the SCIM Servlet Extension*." For custom servlet extensions created using the Server SDK, the process varies based on whether you are using a Java-based or Groovy-scripted extension.

Web Application Servlet Extensions

A Web application may be deployed either as a WAR file that has been packaged according to the standard layout and containing a web.xml deployment descriptor, or from a directory containing the application's source components arranged in the standard layout.

When deploying a Web application from a directory, you may specify the location of the web.xml deployment descriptor if it is not in the standard location. You may also specify the directory used by the server for temporary files. At runtime the web application has access to the server classes.

Java-based Servlet Extensions

For Java-based extensions, first use the Server SDK to create and build the extension bundle as described in the Server SDK documentation. Then, install it using the `manage-extension` tool as follows:

```
$ bin/manage-extension --install/path/to/extension.zip
```

The Java-based extension may then be configured for use in the server using `dsconfig` or the Administrative Console. Create a new Third Party HTTP Servlet Extension, specifying the fully-qualified name for the `HTTPServletExtension` subclass in the `extension-class` property, and providing any appropriate arguments in the `extension-argument` property.



Note: Web Application and Servlet extensions run in a shared embedded web application server environment. Incompatibilities or conflicts may arise from use of different versions of commonly used jars or including frameworks such as loggers, Spring components, JAX-RS implementations or other resources which may require a dedicated Java Virtual Machine (JVM) environment. After introducing a custom extension, check the server error log for an indication that the extension loaded successfully. The error log may also contain debug information if the extension failed to load with an initialization exception or did not complete initialization.

Groovy-Scripted Extensions

For Groovy-scripted extensions, place the necessary Groovy scripts in the appropriate directory (based on the package for those scripts) below the `lib/groovy-scripted-extensions` directory. Then, create a new Groovy Scripted HTTP Servlet Extension, specifying the fully-qualified Groovy class name for the `script-class` property, and providing any appropriate arguments in the `script-argument` property.

Configuring HTTP Operation Loggers

Servlet extensions may write error log messages in the same way as any other kind of server component, but interaction with HTTP clients will not be recorded in the server access log. However, if a servlet extension performs internal operations to interact with data held in the directory server, then those operations may be captured in the access log. To capture information about communication with HTTP clients, you must configure one or more HTTP operations loggers.

By default, the server comes with a single HTTP operation logger implementation, which uses the standard W3C common log format. It records messages in a format like the following:

```
127.0.0.1 - - [01/Jan/2012:00:00:00 -0600]"GET/hello HTTP/1.1" 200 113
```

The log message contains the following elements:

- The IP address of the client that issued the request.
- The RFC 1413 (ident) identity of the client. Because the ident protocol is not typically provided by HTTP clients, the HTTP connection handler never requests this information. This identity will always be represented as a dash to indicate that information is not available.
- The authenticated identity determined for the request by HTTP authentication, or a dash to indicate that the request was not authenticated.
- The time that the request was received.
- The request issued by the client, including the HTTP method, path and optional query string, and the HTTP protocol version used.
- The integer representation of the HTTP status code for the response to the client.
- The number of bytes included in the body of the response to the client.

To configure an HTTP operation logger to use this common log format, create a new instance of a Common Log File HTTP Operation Log Publisher object, specifying the path and name for the active log file to be written and the rotation and retention policies that should be used to manage the log files. In general, properties for Common Log File HTTP Operation Log Publisher objects have the same meaning and use as they do for other kinds of loggers.

You can use the Server SDK to create custom Java-based or Groovy-scripted HTTP operation loggers using the Third Party HTTP Operation Log Publisher and Groovy Scripted HTTP Operation Log Publisher object types.

Example HTTP Log Publishers

When troubleshooting HTTP Connection Handler issues, administrators should first look at the logs to determine any potential problems. The following section shows some `dsconfig` commands and their corresponding log files.

Default Configuration Example. You can configure a default detailed HTTP Log Publisher with default log rotation and retention policies as follows:

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

The corresponding log file provides access information below. The log message contains the following elements:

- The time that the request was received.
- The request ID issued by the client, including the IP address, port, HTTP method, and URL.
- The authorization type, request content type, and status code.
- The response content length.

- The redirect URI.
- The response content type.

The HTTP log file is shown as follows:

```
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300604 from="10.5.1.10:53269"
method="GET" url="https://10.5.1.129:443/Gimel/Users/uid=user.402914,ou=People,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.145 responseContentLength=1530 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.402914,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300605 from="10.5.1.10:53269"
method="PUT" url="https://10.5.1.129:443/Gimel/Users/uid=user.207585,ou=people,
dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.872 responseContentLength=1532 redirectURI="https://
x2270-11.example.lab:443/Gimel/Users/uid=user.207585,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:11:31:18 -0600] RESULT requestID=4309872 from="10.5.1.10:3
```

Configuration with Request/Response Header Names and Values. The following example adds request/response header names and values, including the "Content-Type" request header, which is normally suppressed by default.

```
$ bin/dsconfig set-log-publisher-prop \
--publisher-name "HTTP Detailed Access Logger" \
--set log-request-headers:header-names-and-values \
--remove suppressed-request-header-name:Content-Type \
--set log-response-headers:header-names-and-values
```

The following is a log example of a query request by a SCIM Server using the property, `scim-query-rate`. The log message contains the basic log elements shown in the first example plus the following additional information:

- The request header for the host, http method, content type, connection, user agent.
- The response header for the access-control credentials.

```
[23/Oct/2012:11:39:41-0600] RESULT requestID=4665307 from="10.5.0.20:56044"
method="GET" url="https://10.5.1.129:443/Beth/Users?attributes=username,title,
emails,urn:scim:schemas:extension:custom:1.0:descriptions,urn:scim:schemas:
extension:enterprise:1.0:manager,groups,urn:scim:schemas:extension:custom:1.0:
blob&filter=username+eq+%22user.18935%22" requestHeader="Host: x2270-11.example.
lab:443" requestHeader="Accept: application/json" requestHeader="Content-Type:
application/json" requestHeader="Connection: keep-alive"
requestHeader="User-Agent: Wink Client v1.1.2" authorizationType="Basic"
requestContentType="application/json" statusCode=200 etime=140.384
responseContentLength=11778 responseHeader="Access-Control-Allow-Credentials:
true" responseContentType="application/json"
```

Another log example shows an example user creation event. The client is curl.

```
[23/Oct/2016:11:50:11-0600] RESULT requestID=4802791 from="10.8.1.229:52357"
method="POST" url="https://10.2.1.113:443/Aleph/Users/" requestHeader="Host: x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept: applica-
tion/xml" requestHeader="Content-Type: application/xml" requestHeader="User-Agent:
curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r zlib/1.2.5"
authorizationType="Basic" requestContentType="application/xml" requestContent-
Length=1773 statusCode=201 etime=11.598 responseContentLength=1472 redirec-
tURI="https://x2270-11.example.lab:443/Aleph/Users/b2cef63c-5e46-11e1-974b-
60334b1a0d7a" responseContentType="application/xml"
```

The final example shows a user deletion request. The client is the Sync Server.

```
[23/Oct/2016:11:38:06-0600] RESULT requestID=4610261 from="10.5.1.114:34558"
method="DELETE" url="https://10.2.1.113:443/Aleph/Users/b8547525-24e0-41ae-b66b-
0b441800de70" requestHeader="Host: x2270-11.example.lab:443" requestHeader="Accept:
application/json" requestHeader="Content-Type: application/json" requestHeader="Con-
nection: keep-alive" requestHeader="User-Agent: PingDataSync-6.0.0.0 (Build
20121022173845Z, Revision 11281)" authorizationType="Basic" requestContentType="appli-
cation/json" statusCode=200 etime=10.615 responseContentLength=0
```


Configuring HTTP Connection Handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. They can also be used to host web applications on the server. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Server from starting. The Directory Server's `start-ds` tool will output any errors to the error log. This allows the Directory Server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with an HTTP connection handler include:

- **listen-address.** Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.
- **listen-port.** Specifies the port on which the connection handler will listen for requests from clients. Required.
- **use-ssl.** Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.
- **http-servlet-extension.** Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.
- **http-operation-log-publisher.** Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.
- **key-manager-provider.** Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.
- **trust-manager-provider.** Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.
- **num-request-handlers.** Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.
- **web-application-extension.** Specifies the Web applications to be hosted by the server.

To Configure an HTTP Connection Handler

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, an HTTP log publisher. The HTTP servlet extension and

log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

1. The first step is to configure your HTTP servlet extensions. The following example uses the `ExampleHTTPServletExtension` in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \  
  --extension-name "Hello World Servlet" \  
  --type third-party \  
  --set "extension-  
class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \  
  --set "extension-argument:path=/" \  
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

```
$ bin/dsconfig create-log-publisher \  
  --publisher-name "HTTP Common Access Logger" \  
  --type common-log-file-http-operation \  
  --set enabled:true \  
  --set log-file:logs/http-common-access \  
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \  
  --set "rotation-policy:Size Limit Rotation Policy" \  
  --set "retention-policy:File Count Retention Policy" \  
  --set "retention-policy:Free Disk Space Retention Policy"  
  
$ bin/dsconfig create-log-publisher \  
  --publisher-name "HTTP Detailed Access Logger" \  
  --type detailed-http-operation \  
  --set enabled:true \  
  --set log-file:logs/http-detailed-access \  
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \  
  --set "rotation-policy:Size Limit Rotation Policy" \  
  --set "retention-policy:File Count Retention Policy" \  
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like `listen-port`, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \  
  --handler-name "Hello World HTTP Connection Handler" \  
  --type http \  
  --set enabled:true \  
  --set listen-port:8443 \  
  --set use-ssl:true \  
  --set "http-servlet-extension:Hello World Servlet" \  
  --set "http-operation-log-publisher:HTTP Common Access Logger" \  
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \  
  --set "key-manager-provider:JKS" \  
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to `TRUE` by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \  
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

To Configure an HTTP Connection Handler for Web Applications

1. Create the Web application servlet extension.

```
$ bin/dsconfig create-web-application-extension \  
--extension-name "Hello Web Application" \  
--set "base-context-path:/hello-app" \  
--set "document-root-directory:/opt/hello-web-app"
```

2. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to `TRUE` by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \  
"(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

Domain Name Service (DNS) Caching

If needed, two global configuration properties can be used to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

- **network-address-cache-ttl** – Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
- **network-address-outage-cache-enabled** – Caches hostname-to-IP address results in the event of a DNS outage. This is set to true by default, meaning name resolution results are cached. Unexpected service interruptions may occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP Address Reverse Name Lookups

PingData servers do not explicitly perform numeric IP address-to-hostname lookups. However, address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- `ACI dns`: bind rules under *Managing Access Control* (Directory Server and Directory Proxy Server)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (Directory Server)

- Connection Criteria: *Restricting Server Access Based on Client IP Address* (Directory Server and Directory Proxy Server)
- Connection Handlers: restrict server access using Connection Handlers (Configuration Reference Guide for all servers)

Configuring Traffic Through a Load Balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, then it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address must be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and PingData server. See the product documentation for the external device type. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP), and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

Working with the Referential Integrity Plug-in

Referential integrity is a plug-in mechanism that maintains the DN references between an entry and a group member attribute. For example, if you have a group entry consisting of member attributes specifying the DNs of printers, you can enable the referential integrity plug-in to ensure that the group entry is automatically removed if a printer entry is removed from the Directory Server.

The Referential Integrity Plug-in is disabled by default. When enabled, the plug-in performs integrity updates on the specified attributes (for example, member or uniquemember) after a delete, modify DN, or a rename (i.e., subordinate modifyDN) operation is logged to the `logs/referint` file. If an entry is deleted, the plug-in checks the log file and makes the corresponding change to the associated group entry.

Three important points about the Referential Integrity Plug-in:

- All specified attributes that are configured for Referential Integrity must be indexed.
- On replicated servers, the Referential Integrity Plug-in configuration is not propagated to other replicas; therefore, you must manually enable the plug-in on each replica.
- The plug-in settings must also be identical on all machines.
- Subtree delete operations are not allowed if the referential integrity plugin is enabled and configured to operate in synchronous mode. It must be configured to operate in

asynchronous mode (by specifying a nonzero update interval) if subtree delete operations will be performed.

To Enable the Referential Integrity Plug-in

1. Determine the attributes needed for your system. By default, the `member` and the `uniquemember` attributes are set for the plug-in.
2. Run the `dsconfig` tool to enable the Referential Integrity Plug-in.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \  
--set enabled:true
```

Working with the Unique Attribute Plug-in

The Unique Attribute plug-in is used to enforce uniqueness constraints on the values of one or more attributes across a portion of the Directory Server. The plug-in checks for uniqueness prior to an add, modify, or modify DN request and will instruct the server to reject the request if a constraint violation is found.

The plug-in is disabled by default as it can affect performance in heavy write load environments. Once the plug-in is enabled, it does not check for attribute uniqueness on existing entries, but only on new ADD, MODIFY, or MODDN operations. However, administrators can use the `identify-unique-attribute-conflicts` tool to ensure that no such conflicts exist in the data.



Important: All attributes for which uniqueness should be enforced should be indexed for equality in all backends.

Attribute uniqueness can be enforced in replicated environments in which each replica contains the complete set of data for which to provide uniqueness, regardless of whether clients communicate directly with the server or interact with it through a Directory Proxy Server. In such environments, all servers should have identical uniqueness configurations. Note that it is not possible to *completely* prevent conflicts that arise from simultaneous writes on separate replicas. However, such conflicts will be detected after the changes have been replicated and will trigger administrative alert notifications.

For proxied environments that do not have the complete set of data on all servers (e.g., environments that use entry balancing or that store different portions of the DIT on different servers), you can implement the Global Uniqueness Attribute Plug-in on the Directory Proxy Server, instead of enabling the attribute uniqueness plug-in on the Directory Server. For more information, see the Ping Identity Directory Proxy Server Administration Guide.

To Enable the Unique Attribute Plug-in

1. Determine which attributes must be unique in your data.

2. Run the `dsconfig` tool to enable the plug-in. By default, the plug-in type property is set to `postsynchronizationadd`, `postsynchronizationmodify`, `postsynchronizationmodifydn`, `preoperationadd`, `preoperationmodify`, and `preoperationmodifydn`. If you want to set one plug-in type, use the `--set plugin-type:<operation-type>` option. For example, use `--set plugin-type:preoperationadd` with the following command if you only want to check for attribute uniqueness prior to ADD operation.

```
$ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" \  
--set enabled:true
```

Configuring Uniqueness Across Attribute Sets

Attribute uniqueness can be configured across a set of attributes using the `multiple-attribute-behavior` property. The `multiple-attribute-behavior` property can take the following values:

- **unique-within-each-attribute** - If multiple attributes are specified, then uniqueness will be enforced for all values of each attribute, but the same value may appear in different attributes (in the same entry or in different entries). For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-within-each-attribute"` and add:
 - An entry with a `telephoneNumber` value that matches the `telephoneNumber` attribute in the existing entry, then the add request will fail.
 - An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.
 - An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., `123-456-7893`) but differ from the values in the existing entry, then the add request will succeed.
- **unique-across-all-attributes-including-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any other of the listed attributes in the same entry, nor in any of the listed attributes in other entries. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-including-in-same-entry"` and add:
 - An entry with a `telephoneNumber` value (e.g., `123-456-7890`) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.
 - An entry with a `mobile` value that matches the `mobile` attribute in an existing entry, then that too will fail.
 - An entry with a `mobile` value (e.g., `123-456-7890`) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.
 - An entry with a `telephoneNumber` value (e.g., `123-456-7891`) that matches the `mobile` attribute in an existing entry, then that too will fail.

- An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will fail.
- **unique-across-all-attributes-except-in-same-entry** - If multiple attributes are specified, then uniqueness will be enforced across all of those attributes, so that if a value appears in one of those attributes, that value may not be present in any of the listed attributes in other entries. However, the same value may appear in multiple attributes in the same entry. For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-except-in-same-entry"` and add:
 - An entry with a `telephoneNumber` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then the add request will fail.
 - An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then that too will fail.
 - An entry with a `mobile` value (e.g., 123-456-7890) that matches the `telephoneNumber` attribute in an existing entry, then that will fail.
 - An entry with a `telephoneNumber` value (e.g., 123-456-7891) that matches the `mobile` attribute in an existing entry, then that will fail.
 - An entry with the same `telephoneNumber` and `mobile` attribute values (e.g., 123-456-7893) but differ from the values in an existing entry, then the add request will succeed.

To Enable Uniqueness Across Attribute Sets

- Use `dsconfig` to configure the UID Unique Attribute plug-in to apply across multiple attributes. The `multiple-attribute-behavior` property is set to `"unique-within-each-attribute"`, which indicates that uniqueness will be enforced for all values of each attribute (e.g., `telephoneNumber=123-456-7890` and `mobile=123-456-7891`), but the same value (e.g., either 123-456-7890 or 123-456-7891) may appear in different attributes in the same entry or in different entries.

```
$ dsconfig create-plugin \
--plugin "Unique telephoneNumber and mobile" \
--type "unique-attribute" \
--set "enabled:true" \
--set "type:telephoneNumber" \
--set "type:mobile" \
--set "multiple-attribute-behavior:unique-within-each-attribute" \
--no-prompt
```

Working with the Last Access Time Plug-In

The Last Access Time plug-in is used to record the timestamp of the last activity targeting an entry. The plug-in updates the `ds-last-access-time` attribute of the entry when it is accessed by an add, bind, compare, modify, modify DN, or search operation.

The plug-in can be used with the Directory Server Uncached Attribute Criteria, or any application that needs to determine how recently an entry has been accessed. The plug-in also

enables defining request criteria to limit the scope of tracking the last access time. The `max-search-result-entries-to-update` property also prevents mass updates of `ds-last-access-time` when searches contain many results, but may not reflect end-user access. Consider the following when using this plug-in:

- The plugin should be enabled on all servers that have the same configuration.
- An updated `ds-last-access-time` attribute value is replicated like any other change to an entry.
- The `ds-last-access-time` attribute is not returned from a search, unless included in the attributes list explicitly, or given the "+" specification for operational attributes.
- The `ds-last-access-time` value format is `yyyyMMddHHmmss.SSS'Z'`, which provides millisecond-level accuracy, such as `20131207144135.821Z`.
- The `ds-last-access-time` attribute can be indexed with a local database index. The ordering index type is the most relevant, but may require a higher index entry limit (default is 4000) to accommodate searches for entries that have not been accessed in a long period of time. The ordering index type, with a short time range or high index entry limit, will result in indexed search results for requests such as `(ds-last-access-time>=20131207144135.821Z)`.



Important: Deployments prior to version 4.5 using the last access time plug-in should disable the plug-in before upgrading, and then re-enable the plug-in once the update is complete. If servers are running different versions, the `last-access-time` updates may occur with a higher frequency than intended.

Working with the Pass Through Authentication Plug-In

The Pass Through Authentication plugin is used to delegate bind operations to remote LDAP servers by forwarding simple bind requests to an external LDAP server, including Active Directory. The plugin can be configured to attempt a local bind, set or update a local password, and bypass local password policies to ensure remote passwords are migrated.

Consider the following when using this plugin:

- The plugin should be enabled on all servers that have the same configuration.
- Remote servers accepting a forwarded bind request may require connection security, such as a secure StartTLS or LDAPS TLS connection.
- Carefully consider how password changes and password resets are handled. Updating a password in the Directory Server may result in divergent passwords between the local and remote server. The Data Sync Server can be used to synchronize passwords between servers, if needed.

- The plugin only updates local passwords if the forwarded simple bind is successful. Expired passwords on a remote server may return an invalid credentials error causing the overall bind operation to fail.
- Multiple remote servers can be specified. The `server-access-mode` property determines if the servers are accessed in `round-robin`, `failover-on-unavailable`, or `failover-on-any-failure` modes. The default server access mode is `round-robin`.
- The `update-local-password` property indicates whether the local password value should be updated to the value used in the bind request, in the event that the local bind fails but the forwarded bind succeeds. A local entry must previously exist in order to update passwords.
- The `allow-lax-pass-through-authentication-passwords` property indicates whether updates to the local password value should accept passwords that do not meet local password policy requirements.
- The `connection-criteria` property specifies a set of connection criteria that must match the client associated with the local bind request for the bind to be forwarded to the remote server.
- The `request-criteria` property specifies a set of request criteria that must match the local bind request or a local target entry for the bind to be forwarded to the remote server.
- The `dn-map` property specifies one or more DN mappings that can be used to transform bind DNs before attempting to forward the bind to remote servers.
- The `search-base-dn` property is used when searching for a remote user entry using a filter constructed from the pattern defined in the `search-filter-pattern` property. It is not possible to configure both a DN map and search filter pattern. If neither a DN map nor a search filter pattern is defined, then user entries are expected to have the same DN in the local server and the remote servers.

Supporting Unindexed Search Requests

By default, the Directory Server denies all unindexed search requests, except for those issued by the bind DNs that have the `unindexed-search` privilege. This default behavior keeps the server from tying up worker threads on time-consuming, unindexed searches. However, you can turn off the enforcement of the `unindexed-search` privilege to allow any client to perform an unindexed search. To do this, set the `disabled-privilege` global configuration property to `unindexed-search` as follows:

```
$ bin/dsconfig set-global-configuration-prop \  
--set disabled-privilege:unindexed-search
```

If you choose to allow unindexed searches, you may want to cap the maximum number of concurrent unindexed search requests using the `maximum-concurrent-unindexed-searches` global configuration property. You configure this property using `dsconfig` as follows:

```
$ bin/dsconfig set-global-configuration-prop \  
--set maximum-concurrent-unindexed-searches:2
```

You can limit unindexed search privileges for particular clients using the `allow-unindexedsearches` property of the Client Connection Policy. For more information about configuring Client Connection Policies, see “Configuring Client Connection Policies”.

Sun/Oracle Compatibility

For companies that are migrating from a Sun/Oracle server to the Ping Identity Directory Server, the Ping Identity Directory Server provides a `dsconfig` batch file, `sun-ds-compatibility.dsconfig`, which describes the various components that can be configured to make the server exhibit behavior closer to a Sun/Oracle configuration.

Administrators can use the `sun-ds-compatibility.dsconfig` batch file to apply the Directory Server's configuration with the necessary `dsconfig` commands. Simply uncomment the example commands listed in the file, and then run the `dsconfig` command specifying the batch file. Note that this batch file is not comprehensive and must be used together with the `migrate-sun-ds-config` tool, located in the `bin` folder (or `bat` folder for Windows systems) during the migration process. Both the tool and the batch file overlap in some areas but provide good initial migration support from the Sun/Oracle server to an PingData server.

Another useful tool is the `migrate-ldap-schema` tool in the `bin` folder (or `bat` folder for Windows systems), which migrates schema information from an existing LDAP server onto this instance. All attribute type and object class definitions that are contained in the source LDAP server will be added to the targeted instance or written to a schema file.

To Configure the Directory Server for Sun/Oracle Compatibility

1. From the Directory Server server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder. You can use a text editor to view the file.
2. Read the file completely.
3. Apply any changes to the file by removing the comment symbol at any `dsconfig` command example, and then applying the `dsconfig` command specifying the batch file.

```
$ bin/dsconfig --no-prompt --bindDN "cn=Directory Manager" \  
  --bindPassword "password" --batch-file /path/to/dsconfig/file
```

4. Run the `migrate-ldap-schema` tool to move the schema definitions on the source server to the destination PingData server.

```
$ bin/migrate-ldap-schema
```

5. Next, run the `migrate-sun-ds-config` tool to see what differences exist in the PingData configuration versus the Sun/Oracle configuration. On the Ping Identity Directory Server, run the following command:

```
$ bin/migrate-sun-ds-config
```

6. Test the server instance for further settings that may not have been set with the batch file, the `migrate-ldap-schema` tool, or the `migrate-sun-ds-config` tool.

7. If you notice continued variances in your configuration, contact your authorized support provider.

Chapter

7

Configuring Soft Deletes

The Ping Identity Directory Server (version 3.2.4 or later) supports a *soft-delete* feature that preserves a deleted entry's attribute and uniqueness characteristics to allow it to be undeleted or permanently removed at a later date.

This chapter introduces the following topics:

Topics:

- [About Soft Deletes](#)
- [General Tips on Soft Deletes](#)
- [Configuring Soft Deletes on the Server](#)
- [Searching for Soft Deletes](#)
- [Undeleting a Soft-Deleted Entry Using the Same RDN](#)
- [Undeleting a Soft-Deleted Entry Using a New RDN](#)
- [Modifying a Soft-Deleted Entry](#)
- [Hard Deleting a Soft-Deleted Entry](#)
- [Disabling Soft Deletes as a Global Configuration](#)
- [Configuring Soft Deletes by Connection Criteria](#)
- [Configuring Soft Deletes by Request Criteria](#)
- [Configuring Soft Delete Automatic Purging](#)
- [Summary of Soft and Hard Delete Processed](#)
- [Summary of Soft Delete Controls and Tool Options](#)
- [Monitoring Soft Deletes](#)

About Soft Deletes

The standard implementation of an LDAP server allows for adding, renaming, modifying, searching, comparing, and deleting one or more entries. The DELETE operation is, by specification, a destructive operation that permanently removes an entry and its attributes in a Directory Information Tree (DIT) but records the changes in access, and optionally, audit and change logs. During the DELETE operation, any associations such as references and group memberships are severed to reflect the entry that is removed. Meta attributes like operational attributes, which may be unique to an entry like `entryUUID`, will be lost or be different if the same entry is re-added to the Directory Server.

There are cases, however, where a company may want to preserve their deleted entries to allow for possible undeletion at a later date. For example, a company may want to retain account and subscriber entries for their users (e.g., customers, employees, or partners) who leave but later rejoin. Artifacts that a user creates such as account histories, web pages, notes, may be tracked and recovered while a user is deleted or when the user returns as an active customer.

To facilitate this use case, the Directory Server supports a feature called *soft deletes*, which preserves a deleted entry's attributes and entry uniqueness characteristics to allow the entry to be undeleted or permanently removed at a later date. A delete request may result in a soft delete either by the client explicitly requesting a soft delete or by the request matching criteria defined in an active soft delete policy. The soft-deleted entries are renamed by prefixing an `entryUUID` operational attribute to the DN and adding an auxiliary object class, `ds-soft-delete-entry`, to the entry, which saves the entry in a hidden state. All active references and group memberships are then removed. Once in this hidden state, soft-deleted entries are inaccessible to clients under normal operating conditions. Only clients with the `soft-delete-read` privilege will be allowed to interact with soft-deleted entries.

To allow soft deletes, the Directory Server's attribute uniqueness function has been relaxed to allow for the co-existence of a soft-deleted entry and an active entry with identical naming attributes, such as `uid`. For example, if a user John Smith was soft deleted but a different John Smith was added to the user accounts system, both entries could reside in the DIT without conflict: one in a soft-deleted state; the other, in an active state. The Directory Server extends this capability further by allowing multiple users with the same DN, who would normally conflict if active, to reside in the soft-deleted state.

Soft-deleted entries can be restored with an Undelete operation. However, the same uniqueness constraints that apply when adding a new user to the Directory Server are enforced when a soft-deleted entry is undeleted. Returning to the previous example, John Smith was soft deleted but a different John Smith with the same `uid` as the original John Smith was added later to the system. If the original John Smith was undeleted from its soft-deleted state, it would result in a conflict with the active John Smith entry. Administrators will need to modify the DN of the soft-deleted entry to avoid such conflicts.

Administrators can permanently remove a soft-deleted entry by performing a regular DELETE operation on it. This operation, called a *hard delete*, permanently removes a soft-deleted entry from the server. Also, note that you can also permanently remove a regular non-soft-deleted entry using a hard delete. This is useful when the server is configured with a soft-delete policy that would otherwise turn a regular delete request into a soft delete.

The Directory Server provides tool arguments that can use the Soft Delete Request Control, a Hard Delete Request Control, and other controls necessary to process these operations. Procedures to show how to use these options are presented later in this section.

For replicated topologies, when a participating directory server soft deletes an entry, it notifies the other replicas in the topology to soft delete the same entry on its respective machine. The changelog backend also records these entries by annotating them using an attribute that indicates its soft-deleted state. Modification and hard deletes of soft-deleted entries are not recorded by default in the changelog but can be enabled in the server. For maximum compatibility, it is highly recommended that all servers in the replication cluster support Soft Deletes and have identical Soft Delete configurations.

General Tips on Soft Deletes

There are some general tips about soft deletes that administrators should be aware of:

- **LDAP SDK and Server SDK.** The LDAP SDK and Server SDK both fully supports soft-deletes.
- **Possible Performance Impact for Searching Regular Entries and Soft-Deleted Entries.** There is little performance difference between retrieving a regular entry and a soft-deleted entry, respectively. However, there may be a performance impact when a search operation has to match criteria (such as, `uid=john.smith`) for both active entries and soft-deleted entries. For example, if there is one active `uid=john.smith` entry and two soft-deleted `uid=john.smith` entries, it may take the server a little more time to retrieve and try to match the criteria before it can return the results.
- **Soft Delete for Uncached Attributes and Entries.** The soft delete feature fully supports uncached attributes and uncached entries. See the section on [Uncached Attributes and Entries](#) for more information about the feature.
- **Soft Delete for Leaf Nodes Only.** Soft-deletion of any parent entry is not allowed. Likewise, soft-deleted entries that have soft-deleted sub-entries are not allowed.
- **Attempting to Soft-Delete a Soft-Deleted Entry Fails.** There are two available state options for soft-deletes: administrators can permanently delete a soft-deleted entry or undelete the entry. An administrator cannot soft-delete an already soft-deleted entry, which returns an `UNWILLING_TO_PERFORM` result code.
- **Soft-Deleted Users Have No Privileges.** Soft-deleted users do not have the ability to bind to the Directory Server or have authentication access. They cannot change their passwords and cannot undelete themselves. Also, soft-deleted entries cannot be used as an authorization identity using the proxied authorization or immediate client control. It is important to note that the soft-delete process does not destroy privilege assignment. If a soft-deleted entry is undeleted, the restored entry will retain the same privileges it originally had before being soft deleted. (One possible exception to this are those privileges assigned by virtual attributes that no longer match the newly-undeleted entry; those entries do not retain their original privileges.)
- **Soft-Deleted Entries Not Accessible by Other Means.** Soft-deleted entries may not be accessible from alternate access methods like SCIM.

- **Soft-Deleted Entries Can Be Modified but Not Renamed.** Administrators can search for all soft-deleted entries and the original source entry attributes can be updated as long as the administrators has modify privileges and access to the `soft-delete-read` privilege. Any attempt to rename a soft-deleted entry using a MODIFY DN operation will result in an `UNWILLING_TO_PERFORM` result code.
- **Replication.** Replication will have access to the LDAP operations with Soft Delete controls. These operations are transmitted, processed, and replayed as high-level requests, which are re-played on remote replicated servers. The replication conflict-resolution mechanism handles soft-deleted entries like any regular entries. For example, if a soft delete is executed independently on two servers then replicated, this results in a replication conflict. For maximum compatibility, it is highly recommended that all servers in a replication cluster support Soft Deletes and have identical Soft Delete configuration.
- **Transactions.** Soft-deletes are supported in transactions. The processing workflow uses the transactions mechanism and maintains the context information necessary to rollback failures to soft delete or undelete.
- **Soft-Deletes Through the Directory Proxy Server.** There is no special configuration steps to configure Soft Deletes on the Directory Proxy Server. The soft-deleted entry is routed directly to the underlying Directory Server. There is one exception: in an entry-balancing deployment, the Directory Proxy Server is responsible for routing the soft-deleted entry to the Directory Server containing the originally soft-deleted item. Also, as with standard entry-balanced deployments, it is not possible to undelete (using MODDN) an entry to a different Directory Server.
- **Export-LDIF.** The default behavior is to include soft-deleted entries as part of the `export-ldif` operation. If soft-deleted entries are to be excluded from export, administrators can use the `--excludeSoftDeleteEntries` option to filter out the entries.
- **Proxied Authorization.** The Soft Delete feature can be used with users who have proxied authorization privileges.
- **Ignored by Data Sync Server Sync Pipes.** For customers using the Data Sync Server, soft-deleted entries are not synchronized by the server. Modifications or deletes of a soft-deleted entry are ignored by the Data Sync Server, and do not appear in the changelog by default. An actual soft delete operation appears to the changelog as a regular DELETE, and an actual undelete operation appears in the changelog as a regular ADD.
- **Referential Integrity Plugin Does Not Restore Membership.** References to a deleted DN are not restored by the referential integrity plugin upon undeletion of a soft-deleted entry. For example, if you have referential integrity enabled and you soft-delete a DN that is a member of a static group, the referential integrity plugin will remove this DN from the group's list of members. When you undelete the soft-deleted entry, the plugin will not add the entry back to the group.
- **Criteria-Selected or Explicitly-Requested Purging of Soft Deletes.** The Soft Delete Policy configuration supports two new properties, `soft-delete-retention-time` and `soft-delete-retain-number-of-entries` that performs purging of soft deleted entries. See the section on [Configuring Soft-Delete Automatic Purging](#).
- **Assigning Access to Controls to Non-Root Users Administrators.** By default, the root user account (e.g., `cn=Directory Manager`) has access to all of the controls needed to

run the Soft Delete operations. For non-root users, you must grant access to these Soft Delete controls using access control rules. An example is shown in step 1 of the section, *To Configure Soft Deletes as a Global Configuration*. The following Soft Delete Controls are available to non-root users:

- **Soft Delete Request Control.** Allows the user to perform a soft delete operation. The OID for the control is 1.3.6.1.4.1.30221.2.5.20.
- **Soft Delete Response Control.** Allows the server to hold the DN of the soft-deleted entry that results from a soft delete request. The OID for the control is 1.3.6.1.4.1.30221.2.5.21.
- **Hard Delete Request Control.** Allows the user to run a hard delete operation on the entry, regardless if it is a regular or soft-deleted entry. The OID for the control is 1.3.6.1.4.1.30221.2.5.22.
- **Undelete Request Control.** Allows the user to undelete a soft-deleted entry using an ADD request. The OID for this control is 1.3.6.1.4.1.30221.2.5.23.
- **Soft-Deleted Entry Access Request Control.** Allows the user to search for any soft-deleted entries. The OID for this control is 1.3.6.1.4.1.30221.2.5.24. Note that a bind DN with the `stream-values` privilege can perform operations that can reveal soft-deleted entries, even if that bind DN does not have permission to use the Soft-Deleted Entry Access Request Control. For example, if a user can successfully run `dump-dns` or `ldap-diff`, then that user can get a list of soft-deleted entry DNs or soft-deleted entry contents via the output of one of those tools.

Configuring Soft Deletes on the Server

Soft deletes are configured on the Directory Server in several ways:

- **Using a Soft-Delete Policy and Global Configuration Property.** You can configure soft deletes by creating a soft-delete policy and a global configuration property. The soft-delete policy enables the feature on the server, while the global configuration property sets the controls used for the soft-delete requests. To enter a soft delete request, the `ldapmodify` or `ldapdelete` command requires the `--useSoftDelete` option. A delete request that does not have the `--useSoftDelete` option is treated as a hard delete, which permanently removes the entry.
- **Using Connection Criteria.** You can configure soft deletes by defining connection criteria within a client connection policy. Any clients that meet the criteria will have their deletes processed as soft deletes.
- **Using Request Criteria.** You can configure soft deletes by defining request criteria within a client connection policy. Any client requests that meet the criteria will have their deletes processed as soft deletes. Note that you can define both connection criteria and request criteria. Both criteria are exclusive and can exist within a soft-delete policy. In this case, the connection criteria is processed first, then the request criteria.

Configuring Soft Deletes as a Global Configuration

You can configure the soft delete feature by creating a soft-delete policy and then setting the configuration property on the server. The presence of the soft-delete policy enables the feature

on the server and allows the global configuration property to send the necessary soft-delete requests.

This configuration setting requires that the `--useSoftDelete` option be used together with the `ldapmodify` or `ldapdelete` commands to send the delete using the Soft Delete Request Control. Without the `--useSoftDelete` option, any delete will be processed as a hard delete.

To Configure Soft Deletes as a Global Configuration

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. See steps 1 and 3 in the Configuring Administrator Accounts section for more information. The following LDIF adds ACI's to the `uid=admin,dc=example,dc=com` entry to allow access to user and operational attributes as well as access to the controls necessary to carry out the Soft Delete operations. The LDIF also adds the `soft-delete-read` privilege, which allows the user to access soft-deleted entries. Add the LDIF file using the `ldapmodify` tool.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*|+")(
  (version 3.0; acl "Allow admins to read and write all user and operational
  attributes";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.21")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control and
  Soft Delete Response Control";
  allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.22")
  (version 3.0; acl "Allow admins to use the Hard Delete Request Control";
  allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.23")
  (version 3.0; acl "Allow admins to use the Undelete Request Control";
  allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft-Deleted Entry Access Request
  Control";
  allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
$ bin/ldapmodify --fileName add-admin-aci.ldif
```

2. Create another LDIF file and assign the `soft-delete-read` privilege to the authorized user. Add the file to the server using the `ldapmodify` tool. You'll need to include bind credentials with the `ldapmodify` command.

```
dn: uid=admin,dc=example,dc=com
changeType: modify
add: ds-privilege-name
ds-privilege-name: soft-delete-read
$ bin/ldapmodify --fileName add-admin-priv.ldif
```

3. Configure a soft delete policy using the `dsconfig` command. The soft delete configuration option requires a soft-delete policy, which effectively enables the feature on the server. This is a required step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy
```

4. Configure the soft delete as a global configuration property using the `dsconfig` command. The command sets up the soft-delete controls necessary to send them as a request. This is a required step.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

After a successful modification, the server issues a warning that soft deletes are not enabled and that administrative action may be needed to prune older soft-deleted entries if resources are limited. For more information on setting up soft deletes, see “Configuring Soft Deletes by Connection Criteria”.

One or more configuration property changes require administrative action or confirmation/notification. Those properties include:

```
* soft-delete-policy: Presently the server does not have an automatic purging
  capability for soft deleted entries. The administrator must
  periodically monitor and determine if too many soft deleted entries are
  consuming resources and must be pruned.
```

The Global Configuration was modified successfully

5. Delete an entry to test the soft delete feature. You must use the `--useSoftDelete` (or the short form, `-s`) option with the `ldapdelete` or `ldapmodify` to send the delete using a Soft Delete Request Control. The control renames the entry, prefixes an `entryUUID` operational attribute to the DN, and hides the entry by adding an auxiliary object class, `ds-soft-delete-entry`. If you do not include the `--useSoftDelete` option, the server processes the delete as a hard delete, which permanently removes the entry unless soft deletes is enabled. Upon a successful soft delete operation, the server returns the DN of the newly soft-deleted entry.

```
$ bin/ldapdelete --useSoftDelete uid=user.1,ou=People,dc=example,dc=com
```

```
Processing DELETE request for uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN uid=user.1,ou=People,dc=example,dc=com
Soft-deleted entry DN:
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com
```

Note:



Once an entry has been soft deleted, it is possible for an LDAP client to create an entry with the same DN as the original user entry DN. For example, if you soft-delete an entry (e.g., `uid=user.3,ou=People,dc=example,dc=com`), you can create a new entry using the same DN as the original entry, `uid=user.3,ou=People,dc=example,dc=com`. If the soft-deleted entry is undeleted at a later date, then a conflict will occur.

Searching for Soft Deletes

Soft-deleted entries are excluded from normal LDAP searches because they represent "deleted" entries. The `ldapsearch` tool has been updated to support these types of searches. If you want the option to search for soft-deleted entries, there are three ways to do so:

- **Base-Level Search on a Soft-Deleted entry by DN.** Use `ldapsearch` and specify the base DN of the specific soft-deleted entry that you are searching for.

- **Filtered Search by `soft-delete-entry` object class.** To search for all soft-deleted entries, use `ldapsearch` with a filter on the `ds-soft-delete-entry` objectclass.
- **Soft-Delete-Entry-Access Control.** You can use the Soft Delete Entry Access Control with the LDAP search to return soft-deleted entries. The `ldapsearch` tool provides a shortcut option, `--includeSoftDeletedEntries`, that sends the control to the server for processing. The control allows for the following search possibilities:
 - Return only soft-deleted entries.
 - Return non-deleted entries along with soft-deleted entries.
 - Return only soft-deleted entries in undeleted form.

To Run a Base-Level Search on a Soft-Deleted Entry

- Run `ldapsearch` using the base DN of the specified soft-deleted entry.

```
$ bin/ldapsearch \
  --baseDN entryUUID=4e9b7847-edcb-3791-
  b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com \
  --searchScope base "(objectClass=*)"

# Soft-deleted entry DN:
# entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-soft-delete-entry
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.1
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
```

To Run a Filtered Search by `soft-delete-entry` Object Class

- Run `ldapsearch` to retrieve all soft-deleted entries using the `ds-soft-delete-entry` object class. The following command retrieves all soft-deleted entries on the server

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  "(objectclass=ds-soft-delete-entry)"
```

To Run a Search using the Soft Delete Entry Access Control

The following examples use the `--includeSoftDeletedEntries {with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}` option, which uses the Soft Delete Entry Access Control. You could also use the `--control` option with the Soft Delete Entry Access Control symbolic name, `softdeleteentryaccess`, or the `--control` option with the actual Soft Delete Entry Access Control OID, `1.3.6.1.4.1.30221.2.5.24`.

- **Return Only Soft-Deleted Entries.** Run `ldapsearch` using the `--includeSoftDeletedEntries` option with the value of `without-non-deleted-entries` to return only soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries without-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Non-Deleted Entries Along with Soft-Deleted Entries.** Run `ldapsearch` using the `--includeSoftDeletedEntries` option with the value of `with-non-deleted-entries` to return non-deleted entries along with soft-deleted entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries with-non-deleted-entries \
  --searchScope sub "(objectclass=*)"
```

- **Return Only Soft-Deleted Entries in Undeleted Form.** Run `ldapsearch` using the `--includeSoftDeletedEntries` option with the value of `deleted-entries-in-undeleted-form` to return only soft-deleted entries in undeleted form. Some applications require access to all entries in the server, including both active and soft-deleted entries. The following command returns all entries that were soft-deleted but presents it in a form that is similar to a regular entry with the soft-delete DN in comments. This regular entry format does not show the actual soft-deleted DN but displays it in an "undeleted" form even though it is not actually "undeleted". Also, the object class, `ds-soft-delete-entry`, is not displayed:.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --includeSoftDeletedEntries deleted-entries-in-undeleted-form \
  --searchScope sub "(ds-soft-delete-from-dn=*)"
```

```
# Soft-deleted entry DN:
# entryUUID=2b5511e2-7616-389b-ab0c-025c805ad32c+uid=user.14,ou=People,dc=example,dc=com
dn: uid=user.14,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Abdalla Abdou$78929 Hillcrest Street$Elmira, ME 93080
postalCode: 93080
description: This is the description for Abdalla Abdou.
uid: user.14
userPassword: {SSHA}7GkzWiMiU12m5m+xBV+ZsoX3gVacMcRtSwDTFg==
employeeNumber: 14
initials: AFA
givenName: Abdalla
pager: +1 307 591 4870
mobile: +1 401 069 1289
cn: Abdalla Abdou
sn: Abdou
telephoneNumber: +1 030 505 6190
street: 78929 Hillcrest Street
homePhone: +1 119 487 2328
l: Elmira
mail: user.14@maildomain.net
st: ME
```

Undeleting a Soft-Deleted Entry Using the Same RDN

To undelete a soft-deleted entry, use `ldapmodify` with the `--allowUndelete` option and target the specific soft-deleted entry that you want to restore. In an LDIF file or from the command line, specify the `dn:<target entry>` attribute, which is the DN that the entry will be undeleted *to* and the `ds-undelete-from-dn` attribute, which is the entry that will be undeleted *from*. An undelete requires the `add` changetype so that the entry can be re-added to the server.

To Undelete a Soft-Deleted Entry Using the Same RDN

- Use `ldapmodify` with the `--allowUndelete` option and target the specific soft-deleted entry that you want to restore. The `--allowUndelete` option sends the Soft Undelete Request Control to the server. The first DN is the entry to undelete *to* and the `ds-undelete-from-dn` is the soft-delete entry to undelete *from*.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Undeleting a Soft-Deleted Entry Using a New RDN

In some cases, the original RDN, `uid=user.1`, may have been allocated to a new user, which is allowed while the entry is in a soft-deleted state. To properly undelete this entry, you need to specify a new RDN value that the entry should be restored with. In this case, specifying the RDN of `uid=user.5` will undelete the original entry but with the new DN in the example below. In addition, the `uid` attribute on the entry will be updated with the new value of `user.5` as well. All other attributes of the users entry including the `entryUUID` will remain unchanged.

To Undelete a Soft-Deleted-Entry Using a New RDN

1. Use `ldapmodify` to undelete a soft-deleted entry that has an original RDN, `uid=user.1`, to a new RDN, `uid=user.5`. You will need to ensure that the DN that you are undeleting the entry to does not already exist as this will lead to an "entry already exists" error if you specify a DN that already exists in the Directory Server as a normal entry.

```
$ bin/ldapmodify --allowUndelete
dn: uid=user.5,ou=People,dc=example,dc=com
changetype: add
ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

2. View the results using `ldapsearch`. You will notice the RDN and the `uid` attribute has changed.

```
dn: uid=user.5,ou=People,dc=example,dc=com
objectClass: top
```

```

objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.5
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrfBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4

```

Modifying a Soft-Deleted Entry

You can modify a soft-deleted entry as you would a regular entry using the `ldapmodify` tool and remains hidden in its soft-deleted state after the change. The only restriction is that you cannot change the DN or run a MODDN operation on the soft-deleted entry.

To move a soft-deleted entry from one machine to another, use the `move-subtree` command and specify the DN of the soft-deleted entry. For more information, see [To Move an Entry from One Machine to Another](#).



Note: To modify a soft-deleted entry, the user needs the `soft-delete-read` privilege to access the soft-deleted entry.

To Modify a Soft-Deleted Entry

- Soft-deleted entries can be modified like any regular entry. Use `ldapmodify` and specify the soft-deleted DN.

```

$ bin/ldapmodify
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 390 103 6918
# Processing MODIFY request for entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
# MODIFY operation successful for DN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com

```

Hard Deleting a Soft-Deleted Entry

To permanently remove a soft-deleted entry from the server, you can run `ldapdelete` on the soft-deleted entry for soft-deleted entries. To hard delete a soft-deleted entry, use `ldapdelete` with the `--useHardDelete` option. The Hard Delete Request Control works with soft deletes. It mostly applies when soft delete policies are in place as a means to override soft deletes requests. If soft deletes are configured, running `ldapdelete` with the Hard Delete Request Control (i.e., using the option, `--useHardDelete`) guarantees any entry will be permanently deleted.

To Hard Delete a Soft-Deleted Entry (Global Configuration)

- Run `ldapdelete` on a soft-deleted entry to permanently remove it from the Directory Server. This example assumes that you configured soft deletes as a global configuration for requests.

```
$ bin/ldapdelete \  
  entryUUID=4e9b7847-edcb-3791-  
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com  
  
Processing DELETE request for entryUUID=4e9b7847-edcb-3791-b11b-  
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com  
DELETE operation successful for DN entryUUID=4e9b7847-edcb-3791-b11b-  
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Note:



You cannot soft-delete an already soft-deleted entry. If you use the `--useSoftDelete` with the `ldapdelete` operation on a soft-deleted entry, an error message will be generated.

```
DELETE operation failed.  
Result Code: 53 (Unwilling to Perform)  
Diagnostic Message: DELETE operation failed.
```

To Hard Delete a Soft-Deleted Entry (Connection or Request Criteria)

- Run `ldapdelete` with the `--useHardDelete` option on a soft-deleted entry to permanently remove it from the server. This example assumes that you configured soft deletes using a connection or request criteria.

```
$ bin/ldapdelete --useHardDelete \  
  entryUUID=4e9b7847-edcb-3791-  
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Disabling Soft Deletes as a Global Configuration

To disable soft deletes on your Directory Server, simply reset the global configuration property and the remove the soft-delete policy. From that point, all deletes will be processed as hard

deletes by default. Any use of the soft-deleted options with the LDAP tools results in an error. Any existing soft-deleted entries that are present after the global configuration is disabled will remain in the server as latent entries.

To Disable Soft Deletes as a Global Configuration

1. Run `dsconfig` to reset the global configuration property. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig set-global-configuration-prop --reset soft-delete-policy
```

2. Run `dsconfig` to delete the soft delete policy that you created. Remember to include the LDAP bind parameters for your system.

```
$ bin/dsconfig delete-soft-delete-policy --policy-name default-soft-delete-policy
```

Configuring Soft Deletes by Connection Criteria

The Directory Server supports soft deletes where any delete operation is treated as a soft-delete request as long as the LDAP client meets the connection criteria. You can configure soft deletes by defining the connection criteria used in a client connection policy, and then configuring the soft delete connection criteria in the soft-delete policy.

To Enable Soft Deletes by Connection Criteria

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Create a simple connection criteria using `dsconfig` and name it "Internal Applications". The soft delete connection criteria is configured for a member of a Line of Business (LOB) Applications group connecting from the 10.8.1.0 network.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Internal Applications" \
  --type simple \
  --set included-client-address:10.8.1.0/8 \
  --set "all-included-user-group-dn:cn=LOB Applications,ou=Groups,dc=example,dc=com"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-connection-criteria:Internal Applications"
```

To Disable Soft Deletes by Connection Criteria

- To disable soft deletes by connection criteria, simply reset the `auto-soft-delete-connection-criteria` property on the `soft-delete` policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \  
--policy-name default-soft-delete-policy \  
--reset auto-soft-delete-connection-criteria
```

Configuring Soft Deletes by Request Criteria

Soft deletes can be configured using request criteria within a client connection policy. All delete requests that meet the request criteria is treated as a soft delete. The presence of a soft delete by connection criteria is exclusive of the soft delete by request criteria. Both can be present in a `soft-delete` policy.

To Enable Soft Deletes by Request Criteria

1. Configure a `soft-delete` policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Configure request criteria for soft deletes. The soft delete request criteria is configured for an external delete request from a member of the Internal Applications group matching an entry with object class `inetorgperson` with the request excluding the Soft Delete Request Control and the Hard Delete Request Control.

```
$ bin/dsconfig create-request-criteria \  
--criteria-name "Soft Deletes" \  
--type simple \  
--set "description:Requests for soft delete" \  
--set operation-type:delete \  
--set operation-origin:external-request \  
--set "connection-criteria:Internal Applications" \  
--set not-all-included-request-control:1.3.6.1.4.1.30221.2.5.20 \  
--set "all-included-target-entry-filter:(objectClass=inetorgperson)"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig create-soft-delete-policy \  
--policy-name default-soft-delete-policy \  
--set "auto-soft-delete-request-criteria:Soft Deletes"
```

To Disable Soft Deletes by Request Criteria

- To disable soft deletes by request criteria, reset the `soft-delete` policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \  
--policy-name default-soft-delete-policy \  
--reset auto-soft-delete-request-criteria
```

Configuring Soft Delete Automatic Purging

By default, the Directory Server retains soft-deleted entries indefinitely. For companies that want to set up automatic purging of soft-deleted entries, the server provides two properties on the Soft Delete Policy that can be configured for either the maximum retention time for all soft-deleted entries and/or the retained number of soft-deleted entries. These changes take effect without requiring a server restart.

To Configure Soft-Delete Automatic Purging

You can change either the retention time or the retained number of entries to enable automatic purging. By default, both are set to an indefinite retention time and number of entries. The time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), may be preceded by an integer to specify a quantity for that unit, such as "1 d", "52 w", etc. Once you configure the properties, the changes take effect immediately without the need for a server restart.

Note that the server will delete all of the soft-deleted entries according to the policy in effect. If the policy is changed while entries are in the process of being deleted, the new policy takes effect after the in-process batch of entries is deleted and applies to any remaining soft-deleted entries going forward according to the new policy.

1. Retrieve the name of the Soft Delete Policy in effect using the `dsconfig` command. For this example, the Soft Delete Policy is called `default-soft-delete-policy`.

```
$ bin/dsconfig get-global-configuration-prop \
  --property soft-delete-policy
```

2. Do one or both of the following:

- Run `dsconfig` to set the retention time for soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "soft-delete-retention-time:52 w"
```

- Run `dsconfig` to set the retained number of soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set soft-delete-retain-number-of-entries:1000000
```

3. The Soft Delete Policy must be assigned to the global configuration if it has not been assigned yet.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

To Disable Soft-Delete Automatic Purging

You can disable Soft-Delete automatic purging using the `dsconfig` command. The change takes effect immediately without the need of a server restart. However, if the server is in the middle of an automatic soft-delete purging, it may continue to purge entries until the next time it evaluates the Soft Delete Policy.

- Run `dsconfig` to reset the Soft-Delete Policy properties that control automatic purging: `soft-delete-retention-time` and `soft-delete-retain-number-of-entries`.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset soft-delete-retention-time \
  --reset soft-delete-retain-number-of-entries
```

Summary of Soft and Hard Delete Processed

The following table summarizes the resulting actions of a DELETE operation for soft deletes.

Table 7: If No Automatic Soft Delete Criteria is Configured

Action	Result
Delete	Performs a hard delete on the entry.
Delete with the Soft Delete Request Control	Performs a soft delete on the entry.
Delete of a soft-deleted entry	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Soft Delete Request Control	Not allowed. Generates an UNWILLING_TO_PERFORM error.
Delete with a Hard Delete Request Control	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Hard Delete Request Control.	Performs a hard delete on the entry.

The following table summarizes the resulting actions of a DELETE operation for soft deletes configured by connection criteria or request criteria.

Table 8: If Soft Delete Connection or Request Criteria is Configured

Action	Result
Delete not matching criteria	Performs a hard delete on the entry.
Delete matching criteria	Performs a soft delete on the entry.
Delete of a soft-deleted entry not matching criteria	Performs a hard delete on the entry.

Action	Result
Delete of a soft-deleted entry matching criteria	Performs a hard delete on the entry.
Delete not matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete with Soft and Hard Delete Request Controls	Not allowed. Generates an UNWILLING_TO_PERFORM error.

Summary of Soft Delete Controls and Tool Options

The following table shows the OIDs for each soft delete control. The Soft Delete OIDs are defined in the LDAP SDK generated API documentation.

Table 9: Soft Delete OIDs

OID Type	OID
Soft Delete Request Control	1.3.6.1.4.1.30221.2.5.20
Soft Delete Response Control	1.3.6.1.4.1.30221.2.5.21
Hard Delete Request Control	1.3.6.1.4.1.30221.2.5.22
Soft Undelete Request Control	1.3.6.1.4.1.30221.2.5.23
Soft Delete Entry Access Control	1.3.6.1.4.1.30221.2.5.24

The following table shows the new tool options available for the Soft Delete operations.

Table 10: Soft Delete Tool Options

Action	Result
ldapdelete / ldapmodify	--useSoftDelete/-s . Process DELETE operations with the Soft Delete Request Control, whereby entries are renamed, and hidden instead of being permanently deleted. The Directory Server must be configured to allow soft deletes. Note that any entries in the LDIF file with the changetype of <code>delete</code> will be processed as a soft-delete request.
ldapdelete	--useHardDelete . Process DELETE operations with the Hard Delete Request Control, which bypasses any soft delete policies and processes the delete request immediately without retaining the entry as a soft-deleted entry. The Directory Server must be configured to allow soft deletes.
ldapsearch	--includeSoftDeletedEntries {with-non-deleted-entries without-non-deleted-entries deleted-entries-in-undeleted-form} . Process search operations with the Soft Delete Entry Access Control. Soft delete search options are as follows:

Action	Result
	<ul style="list-style-type: none"> • with-non-deleted-entries. Returns all entries matching the search criteria with the results including non-deleted and soft-deleted entries. • without-non-deleted-entries. Returns only soft-deleted entries matching the search criteria. • deleted-entries-in-undeleted-form. Returns only soft-deleted entries matching the search criteria with the results returned in their undeleted entry form. <p>Users must have access to the Soft Delete Entry Access Control to be able to search for soft-deleted entries.</p>
ldapmodify	<p>--allowUndelete. Process ADD operations which include the <code>ds-undelete-from-dn</code> attribute as undelete requests. Undelete requests re-add previously soft-deleted entries back to the server as non-deleted entries by providing the Undelete Request Control with the ADD operation. The Directory Server must be configured to allow soft deletes to process any undelete requests and the client user must have the <code>soft-delete-read</code> privilege.</p>

The following table shows the symbolic names that can be used with the server's LDAP commands using the `--control/-J` option.

Table 11: Soft Delete OID Symbolic Names using with the `--control/-J` Option

Control	Symbolic Name
Soft Delete Request Control	softdelete
Hard Delete Request Control	harddelete
Soft Undelete Request Control	undelete
Soft Delete Entry Access Control	softdeleteentryaccess

Monitoring Soft Deletes

The Directory Server provides monitoring entries and logs to track all soft delete operations. The access and debug logs do not have any options specific for soft deletes.

New Monitor Entries

Two new monitor entries are present for a backend monitor entry. Administrators will see the following additional monitor entries on `cn=userRoot Backend,cn=monitor:`

- **ds-soft-delete-entry-operations-count.** Displays the number of soft-deletes performed on the backend since server startup.
- **ds-undelete-operations-count.** Displays the number of undeletes performed on the backend since server startup.

- **ds-backend-soft-deleted-entry-count.** Displays the current number of soft-deleted entries in the database.
- **ds-auto-purged-soft-deleted-entry-count.** Displays the current number of soft-deleted entries purged since the backend or server was restarted.

To Monitor Soft Deletes

- Run `ldapsearch` on the `cn=userRoot Backend,cn=monitor` branch. Use a search criteria targeting the `ds-backend-monitor-entry` object class.

```
$ bin/ldapsearch --baseDN "cn=userRoot Backend,cn=monitor" \
  --searchScope sub "(objectclass=ds-backend-monitor-entry)"
```

```
dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
objectClass: extensibleObject
cn: userRoot Backend
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: FALSE
ds-backend-entry-count: 200001
ds-backend-soft-deleted-entry-count: 1000
ds-soft-delete-operations-count: 40
ds-undelete-operations-count: 20
ds-auto-purged-soft-deleted-entry-count: 0
ds-base-dn-entry-count: 200001 dc=example,dc=com
ds-backend-writability-mode: enabled
```

Access Logs

The access log records the LDAP operations corresponding to soft delete and undelete for DELETE, SEARCH, MODIFY, and ADD operations with the related soft-deleted values. The access log does not require any configuration for soft delete.

For DELETE (soft-delete) operations, the access log displays:

```
[14/May/2012:09:40:16.942 -0500] DELETE RESULT conn=18 op=1 msgID=2
dn="uid=user.1,ou=People,dc=example,dc=com" resultCode=0 etime=30.367
softDeleteEntryDN="entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,
ou=People,dc=example,dc=com"
```

For SEARCH operations for soft-deleted entries, the log displays:

```
[14/May/2012:09:40:52.320 -0500] SEARCH RESULT conn=19 op=1 msgID=2
base="dc=example,dc=com" scope=2 filter="(objectclass=ds-soft-delete-entry)"
attrs="ALL" resultCode=0 etime=1.631 entriesReturned=1
```

For MODIFY operations of soft-deleted entries, the log displays:

```
[14/May/2012:09:42:43.679 -0500] MODIFY RESULT conn=20 op=1 msgID=1
dn="entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com" resultCode=0 etime=2.639 changeToSoftDeletedEntry=true
```

For ADD (soft undelete) operations, the log displays:

```
[14/May/2012:09:58:16.728 -0500] ADD RESULT conn=25 op=1 msgID=1
dn="uid=user.0,ou=People,dc=example,dc=com" resultCode=0 etime=22.700
undeleteFromDN="entryUUID=ad55a34a-763f-358f-93f9-da86f9ecd9e4+uid=user.0,
ou=People,dc=example,dc=com"
```

Audit Logs

The audit log captures any MODIFY and DELETE operations of soft-deleted entries. These changes are recorded as fully commented-out audit log entries. The audit log does not require any configuration for soft deletes.

For any soft-deleted entry, the audit log entry displays the `ds-soft-delete-entry-dn` property and its soft-deleted entry DN.

```
# 14/May/2012:10:57:09.054 -0500; conn=30; op=1
# ds-soft-delete-entry-dn: entryUUID=68147342-1f61-3465-8489-
3de58c532130+uid=user.2,ou=People,dc=example,dc=com
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: delete
```

For any MODIFY changes made, the log displays the LDIF, the modifier's name and update time.

```
# 14/May/2012:10:58:33.566 -0500; conn=33; op=1
# dn: entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: modify
# replace: homePhone
# homePhone: +1 003 428 0966
#-
# replace: modifiersName
# modifiersName: uid=admin,dc=example,dc=com
#-
# replace: modifyTimestamp
# modifyTimestamp: 20131010020345.546Z
```

For any undelete of a soft-deleted entry, the log displays the `ds-undelete-from-dn` attribute plus the entry unique ID, create time and creator's name.

```
# 14/May/2012:10:59:21.754 -0500; conn=34; op=1
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: add
uid: user.2
ds-undelete-from-dn: entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=Peo-
ple,dc=example,dc=com
ds-entry-unique-id:: vw1jg801S7GWrTiS3UE5DA==
createTimestamp:: 20131010181148.630Z
creatorsName: uid=admin,dc=example,dc=com
```

For hard (permanent) deletes of a soft-deleted entry, the log displays the soft-deleted entry DN that was removed.

```
# 14/May/2012:11:00:14.055 -0500; conn=36; op=1
# dn: entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: delete
```

To Configure the File-Based Audit Log for Soft Deletes

1. Enable the audit log if it is disabled.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit Logger" \
--set enabled:true
```


- View the audit log. The `soft-delete-entry-audit-behavior` property is set to "commented" by default and provides additional information in comments about the soft-deleted entry that was either created or undeleted.

```
# 11/May/2012:15:33:17.552 -0500; conn=13; op=1
# ds-soft-delete-entry-dn:entryUUID=54716bfd-fbc4-3108-ac37-
bf6b1b166e37+uid=user.15,ou=People,dc=example,dc=com
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: delete
```

Change Log

The change log can be configured to capture soft-delete changes to entries, so that external clients, such as a Data Sync Server, can access these changes. The `ds-soft-delete-entry` attribute represents an entry that has been soft-deleted and is part of the source entry passed into the changelog to indicate the entry has been soft-deleted.

Two important points about soft deletes and the changelog are as follows:

- All soft-delete operations appear in the changelog and appear as a regular DELETE operation. When a soft delete occurs, the resulting changelog entry will include a `ds-soft-delete-entry-dn` operational attribute with the value of the soft-deleted entry DN. If you are using the PingData Data Sync Server, it does recognize the `ds-soft-delete-entry-dn` attribute and does not do anything with it.
- The changelog backend's `soft-delete-entry-included-operation` property determines whether or not MODIFY or DELETE operations of soft-deleted entries appear in the changelog. By default, the property is not enabled by default.

To Configure Soft Deletes on the Changelog Backend

- Configure soft deletes on the changelog backend.

```
$ bin/dsconfig set-backend-prop \
--backend-name changelog \
--set soft-delete-entry-included-operation:delete \
--set soft-delete-entry-included-operation:modify
```

- Run a soft-delete operation on an entry.
- View the changelog for the soft-deleted entry.

```
$ bin/ldapsearch --baseDN cn=changelog \
"(objectclass=*)" "+"

dn: cn=changelog
subschemaSubentry: cn=schema
entryUUID: 9920f7e9-5a04-392a-82a8-32662d7d3863
ds-entry-checksum: 304022441
dn: changeNumber=1,cn=changelog
targetUniqueId: 94f634df-c90e-39aa-bd4a-9183c29746d0
changeTime: 20120511154141Z
ds-soft-delete-entry-dn: entryUUID=94f634df-c90e-39aa-bd4a-
9183c29746d0+uid=user.9,ou=People,dc=example,dc=com
modifyTimestamp: 20131010020345.546Z
createTimestamp:: 20131010181148.630Z
localCSN: 000001373C9008520000000000003
modifiersName: uid=admin,dc=example,dc=com
entry-size-bytes: 298
```

```
subschemaSubentry: cn=schema  
entryUUID: 459b06c6-89f3-307e-a515-22433eb420b6  
createTimestamp: 20120511154141.431Z  
modifyTimestamp: 20120511154141.431Z  
ds-entry-checksum: 1157320579
```

Chapter

8

Importing and Exporting Data

The Ping Identity Directory Server supports import or export of the database backends in LDAP Data Interchange Format (LDIF). The `bin/import-ldif` and `bin/export-ldif` tools can be used to create or export database backends for online or offline servers. The tools support options that can restrict the input or output to a subset of the entries or a subset of the attributes within entries. The tools also provide features to compress, encrypt or digitally sign the data.

This chapter presents the following topics:

Topics:

- [*Importing Data*](#)
- [*Running an Offline Import*](#)
- [*Running an Online LDIF Import*](#)
- [*Adding Entries to an Existing Directory Server*](#)
- [*Filtering Data Import*](#)
- [*Exporting Data*](#)
- [*Encrypting LDIF Exports and Signing LDIF Files*](#)
- [*Filtering Data Exports*](#)
- [*Scrambling Data Files*](#)

Importing Data

The Ping Identity Directory Server provides initialization mechanisms to import database files. The `import-ldif` command-line tool imports data from an LDAP Data Interchange Format (LDIF) file. The data imported by the `import-ldif` command can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the LDIF file. The command also supports importing data that has been compressed, encrypted or digitally signed or both.

The `import-ldif` utility can be run with the server offline or online. If the server is online, administrators can initiate the import from a local or remote client. The LDIF file that contains the import data must exist on the server system. During an online import, the target database repository, or backend, will be removed from service and data held in that backend will not be available to clients.

The `import-ldif` tool has been modified to help guard against accidental overwriting of existing backend data with the addition of the `--overwriteExistingEntries` option. This option must be present when performing an import into a backend with a branch that already contains entries (although the option is not needed if a branch contains just a single base entry).

Validating an LDIF File

Prior to importing data, you can validate an import file using the Directory Server's `validate-ldif` tool. When run, the tool binds to the Directory Server, locally or remotely, and validates the LDIF file to determine whether it violates the server's schema. Those elements that do not conform to the schema will be rejected and written to standard output. You can specify the path to the output file to which the rejected entries are written and the reasons for their rejection. The `validate-ldif` tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

To process large files faster, you can also set the number of threads for validation. The tool also provides options to skip specified schema elements if you are only validating certain items, such as attributes only. Use the `--help` option to view the arguments.

To Validate an LDIF File

- Use the `validate-ldif` tool to validate an LDIF file. Make sure the server is online before running this command.

```
$ bin/validate-ldif --ldifFile /path/to/data.ldif \  
--rejectFile rejectedEntries
```

```
1 of 200 entries (0 percent) were found to be invalid.  
1 undefined attributes were encountered.  
Undefined attribute departmentname was encountered 1 times.
```

Computing Database Cache Estimate

After successful completion of an import, the `import-ldif` command lists detailed information about the database cache characteristics of the imported data set. The current server configuration is considered along with the capabilities of the underlying hardware to guide decisions for changing JVM size and database-cache-percent for the backend.

The `import-ldif` command will complete with a summary of database cache usage characteristics for the imported data set. Additional files are available in the `/logs/tools` directory that describe the database cache characteristics in more detail.

Tracking Skipped and Rejected Entries

During import, entries can be skipped if they do not belong in the specified backend, or if they are part of an excluded base DN or filter. The `--skipFile {path}` argument can be used on the command line to indicate that any entries that are skipped should be written to a specified file. You can add a comment indicating why the entries were skipped.

Similarly, the `--rejectFile {path}` argument can be added to obtain information about which entries were rejected and why. An entry can be rejected if it violates the server's schema constraints, if its parent entry does not exist, if another entry already exists with the same DN, or if it was rejected by a plug-in.

Running an Offline Import

You can run the `import-ldif` tool offline to import LDIF data encoded with the UTF-8 character set. This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated using a MakeLDIF template. You do not need to authenticate as an administrator when performing offline LDIF imports.

To Perform an Offline Import

- Use the `import-ldif` command to import data from an LDIF file. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \  
--rejectFile /path/to/reject.ldif --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a Compressed File

- Use the `import-ldif` command to import data from a compressed gzip formatted file. You must also use the `--isCompressed` option to indicate that the input file is compressed.

Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --isCompressed \  
--ldifFile /path/to/data.gz --rejectFile /path/to/reject.ldif \  
--skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a MakeLDIF Template

- Use the `import-ldif` command to import data from a MakeLDIF template, which is located in the `<server-root>/config/MakeLDIF`. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

The following command uses the standard data template and generates 10,000 sample entries, and then imports the file to the server.

```
$ bin/import-ldif --backendID userRoot \  
--templateFile config/MakeLDIF/example.template
```

Running an Online LDIF Import

Administrators can run LDIF imports while the server is online from another remote server. The online import resembles the offline import, except that you must provide information about how to connect and authenticate to the target server. You can schedule the import of an LDIF file to occur at a particular time using the `--task` and `--start YYYYMMDDhhmmss` options of the `import-ldif` tool.

You can also specify email addresses for users that should be notified whenever the import process completes (regardless of success or failure, or only if the import fails). To set up SMTP notifications, see [Working with the SMTP Account Status Notification Handler](#).

To Perform an Online LDIF Import

- Use the `import-ldif` tool to import data from an LDIF. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server1 --port 389 \  
--bindDN uid=admin,dc=example,dc=com --bindPassword password \  
--backendID userRoot --ldifFile userRoot.ldif
```

To Schedule an Online Import

1. Use the `import-ldif` tool to import data from an LDIF file at a scheduled time. To specify a time in the UTC time zone, include a trailing “Z”. Otherwise, the time will be treated as a local time in the time zone configured on the server. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task \  
--hostname server1 \  
--start YYYYMMDDhhmmssZ
```

```
--port 389 \  
--bindDN uid=admin,dc=example,dc=com \  
--bindPassword password \  
--backendID userRoot \  
--ldifFile /path/to/data.ldif \  
--start 20111026010000 \  
--completionNotify import-complete@example.com \  
--errorNotify import-failed@example.com
```

```
Import task 2011102617321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

2. Confirm that you successfully scheduled your import task using the `manage-tasks` tool to view a summary of all tasks on the system.

```
$ bin/manage-tasks --summary
```

```
ID          Type      Status  
-----  
2011102617321510 Import  Waiting on start time
```

3. Use the `manage-tasks` tool to monitor the progress of this task. Use the task ID of the import task. If you cannot find the task ID, use the `--summary` option to view a list of all tasks scheduled on the Directory Server.

```
$ bin/manage-tasks --info 2011102617321510
```

```
Task          Details  
-----  
ID            2011102617321510  
Type         Import  
Status       Waiting on start time  
Scheduled Start Time  Oct 26, 2011 1:00:00 AM CDT  
Actual Start Time  
Completion Time  
Dependencies Failed  None  
Dependency Action    None  
Email Upon Completion  admin@example.com  
Email Upon Error      admin@example.com  
  
Import      Options  
-----  
LDIF File  /path/to/data.ldif  
Backend ID userRoot
```

To Cancel a Scheduled Import

- Use the `manage-tasks` tool to cancel the scheduled task.

```
$ bin/manage-tasks --cancel 2011102417321510
```

Adding Entries to an Existing Directory Server

To add entries to an existing Directory Server while preserving operational attributes, such as `createTimestamp` or `modifiersName`, the Ignore No User Modification control must be attached to the request. The Ignore No User Modification control allows modification of certain attributes that have the No User Modification constraint. Special care should be used with this control.

The Ignore No User Modification is only applied to ADD requests. Using the control to modify an existing entry, resulting in an operational attribute change, will fail.

To Append Entries to an Existing Directory Server

- Use `ldapmodify` with the Ignore No User Modification control (i.e., the OID is 1.3.6.1.4.1.30221.2.5.5).

```
$ bin/ldapmodify --control 1.3.6.1.4.1.30221.2.5.5 \
--filename change-record.ldif
```

Filtering Data Import

The `import-ldif` command provides a way to either include or exclude specific attributes or entries during an import. The arguments are summarized as follows:

Table 12: Inclusion and Exclusion Arguments for import-ldif

Argument	Description
<code>--includeBranch</code>	Base DN of a branch to include in the LDIF import (can be specified multiple times)
<code>--excludeBranch</code>	Base DN of a branch to exclude from the LDIF import (can be specified multiple times)
<code>--includeAttribute</code>	Attribute to include in the LDIF import (can be specified multiple times)
<code>--excludeAttribute</code>	Attribute to exclude from the LDIF import (can be specified multiple times)
<code>--includeFilter</code>	Search filter to identify entries to include in the LDIF import (can be specified multiple times)
<code>--excludeFilter</code>	Search filter to identify entries to exclude from the LDIF import (can be specified multiple times)
<code>--excludeOperational</code>	Exclude operational attributes from the LDIF import.
<code>--excludeReplication</code>	Exclude replication attributes from the LDIF import.
<code>--excludeSoftDelete</code>	Exclude soft delete entries from the LDIF import.

Exporting Data

The Ping Identity Directory Server `export-ldif` command-line tool exports data from Directory Server backend to an LDAP Data Interchange Format (LDIF) file. The tool must be run in the non-task based mode, which implies that it works outside of the server JVM process. The `export-ldif` must be run without connection or task arguments while the server is either online or offline. This tool exports a point-in-time snapshot of the backend which is guaranteed to provide a consistent state of the database, in LDIF, which can be reimported with `import-ldif` if necessary.

The data exported by `export-ldif` can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the backend. This is accomplished by specifying branches, filters, and attributes to include or exclude. The exported LDIF can be compressed, encrypted or digitally signed.

To Perform an Export

- Use the `export-ldif` command to export data to an LDIF file.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif
```

To Perform an Export from Specific Branches

- Use the `export-ldif` command to export data to an LDIF file under a specific branch from the `userRoot` backend of the local Directory Server into a compressed file. The command also excludes operational attributes from the exported data and wraps long lines at column 80.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif.gz --compress \  
--includeBranch ou=people,dc=example,dc=com --excludeOperational \  
--wrapColumn 80
```

Encrypting LDIF Exports and Signing LDIF Files

The Directory Server provides features to encrypt data during an LDIF export using the `export-ldif --encryptLDIF` option and to allow the encrypted LDIF file to be imported onto the same instance or another server in the same replication topology using the `import-ldif --isEncrypted` option.

The Directory Server also provides an additional argument that digitally signs the contents of the LDIF file, which ensures that the content has not been altered since the export. To digitally sign the contents of the exported LDIF file, use the `export-ldif --sign` option. To allow a signed LDIF file to be imported onto the same instance or another server in the same topology, use the `import-ldif --isSigned` option.

Note that there is not much added benefit to both signing and encrypting the same data, since encrypted data cannot be altered without destroying the ability to decrypt it.

To Encrypt an LDIF Export

- Run `export-ldif` tool with the `--encryptLDIF` option to encrypt the data during an export to an output LDIF file. The following command runs an offline export of the `userRoot` backend, and encrypts the file when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot --ldifFile /path/to/data.ldif \  
--encryptLDIF
```

To Import an Encrypted LDIF File

An encrypted LDIF file can be imported into the same instance from which it was exported, or into any other server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not in some way connected to the instance from which it was exported.

- Run the `import-ldif` tool to import the encrypted LDIF file from the previous example. The command imports the `data.ldif` file, decrypts the contents while overwriting the existing contents to the `userRoot` backend. The command requires the `--isEncrypted` option, which instructs the tool that the LDIF file is encrypted.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isEncrypted
```

To Sign an Export

- Run `export-ldif` tool with the `--sign` option to digitally sign the data during an export to an output LDIF file. The following command runs an offline export of the `userRoot` backend, and signs the content when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif --sign
```

To Import a Signed LDIF File

- Run the `import-ldif` tool to import the signed LDIF file from the previous example. The command imports the `data.ldif` file, checks the signature of the contents while overwriting the existing contents to the `userRoot` backend. The command requires the `--isSigned` option, which instructs the tool that the contents of the LDIF file is signed.

```
$ bin/import-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Filtering Data Exports

The `export-ldif` command analogous arguments to the `import-ldif` tool to provide a way to either include or exclude specific attributes or entries during an export. The arguments are summarized as follows:

Table 13: Inclusion and Exclusion Arguments for export-ldif

Argument	Description
<code>--includeBranch</code>	Base DN of a branch to include in the LDIF export (can be specified multiple times)
<code>--excludeBranch</code>	Base DN of a branch to exclude from the LDIF export (can be specified multiple times)
<code>--includeAttribute</code>	Attribute to include in the LDIF export (can be specified multiple times)

Argument	Description
--excludeAttribute	Attribute to exclude from the LDIF export (can be specified multiple times)
--includeFilter	Filter to identify entries to include in the LDIF export (can be specified multiple times)
--excludeFilter	Filter to identify entries to exclude from the LDIF export (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF export.
--excludeReplication	Exclude replication attributes from the LDIF export.
--excludeSoftDelete	Exclude soft delete entries from the LDIF export.

Scrambling Data Files

The Directory Server `transform-ldif` tool provides backward compatibility with the former `scramble-ldif` tool, with additional functionality for configuring input and output files. The `transform-ldif` tool reads data from one or more source LDIF files and writes the transformed data to a single output file.

Using this tool to scramble data, enables obscuring the values of certain attributes so that it is difficult to determine the original values in the source data, while also preserving the characteristics of the associated attribute syntax. This process is repeatable, so that if the same value appears multiple times, it will yield the same scrambled representation each time. Scrambling can be applied to both LDIF entries and LDIF change records.

The process of scrambling data is not the same as encryption. It should only be used to provide simple obfuscation of data. The following are general guidelines for scrambling attributes:

- If the attribute is `userPassword` and its value starts with a scheme name surrounded by curly braces, such as `{SSHA256}XrgyNdl3fid7KYdhd/Ju47KJQ5PYZqlUlyzxQ28f/QXUnNd9fupj9g==`, the scheme name will be left unchanged and the rest of the value will be treated like a generic string.
- If the attribute is `authPassword` and its value contains at least two dollar signs, such as `"SHA256$QGbHtDCi1i4=$8/X7XRGaFCovC5mn7ATPDYIkVoocDD06Zy3lbd4AoO4="`, the portion up to the first dollar sign (which represents the name of the encoding scheme) is preserved and the remainder of the value is treated like a generic string.
- If an attribute has a Boolean syntax, the scrambled value will be either `TRUE` or `FALSE`. The determination to use a value of `TRUE` or `FALSE` is random, so scrambling Boolean values is not repeatable. By randomizing the scrambling for Boolean values, the syntax and obfuscation of the original value is preserved.
- If an attribute has a distinguished name syntax (or a related syntax, such as a name and optional UID), scrambling is applied to the values of RDN components for any attributes to be scrambled. For example, if the tool is configured to scramble both the `member` and `uid` attributes, and an entry has a `member` attribute with a value of `"uid=john.doe,ou=People,dc=example,dc=com"`, that `member` value will be scrambled in a way that only obscures the "john.doe" portion but leaves the attribute names and all values of non-scrambled attributes intact.
- If an attribute has a generalized time syntax, that value is replaced with a randomized timestamp using the same format (the same number of digits and the same time zone

indicator). The randomization will be over a time range that is double the difference between the time the `transform-ldif` tool was launched and the timestamp to be scrambled. For values where that time difference is less than one day, one day will be added to the difference before it is doubled.

- If an attribute has an integer, numeric string, or telephone number syntax, scrambling is only applied to numeric digits while all other characters are left intact. If there are multiple digits, then the first digit will be nonzero.
- If an attribute has an octet string syntax, it is scrambled as follows:
 - Each byte that represents a lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
 - Each byte that represents an uppercase ASCII letter is replaced with a randomly-selected uppercase ASCII letter.
 - Each byte that represents an ASCII digit is replaced with a randomly-selected ASCII digit.
 - Each byte that represents a printable ASCII symbol is replaced with a randomly-selected printable ASCII symbol.
 - Each byte that represents an ASCII control character is replaced with a randomly-selected ASCII letter, digit, or symbol.
 - Each non-ASCII byte will be replaced with a randomly-selected non-ASCII byte.
- If an attribute has a value that represents a valid JSON object, the resulting value will also be a JSON object. All field names will be left intact, and only the values of those fields may be scrambled. If the `--scrambleJSONfield` argument is provided, only the specified fields will have values scrambled. Otherwise, the values of all fields will be scrambled. Field values are scrambled as follows:
 - Null values are not scrambled.
 - Boolean values are replaced with randomly-selected Boolean values. As with attributes with a Boolean syntax, these values are non-repeatable.
 - Number values will have only their digits replaced with randomly-selected digits and all other characters (minus sign, decimal point, exponentiation indicator) are left unchanged.
 - String values will be replaced with a randomly-selected generic string.
 - Array values have scrambling applied as appropriate for each value in the array. If the array field itself should be scrambled, then all values in the array are scrambled. Otherwise, only JSON objects contained inside the array have scrambling applied to appropriate fields.
 - JSON values have scrambling applied as appropriate for their fields.
- If an attribute does not match any of the previous criteria, it is scrambled as follows:
 - Each lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.

- Each uppercase ASCII letter replaced with a randomly-selected uppercase ASCII letter.
- Each ASCII digit is replaced with a randomly-selected ASCII digit.
- All other characters are left unchanged.

The following example reads from an LDIF file named `original.ldif`, scrambles the values of the `telephoneNumber`, `mobile`, and `homeTelephoneNumber` attributes, and writes the results to `scrambled.ldif`:

```
$ bin/transform-ldif --sourceLDIF original.ldif \  
--targetLDIF scrambled.ldif \  
--scrambleAttribute telephoneNumber \  
--scrambleAttribute mobile \  
--scrambleAttribute homeTelephoneNumber \  
--randomSeed 0
```


Chapter

9 Backing Up and Restoring Data

The Ping Identity Directory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler.

This chapter presents the following topics:

Topics:

- [*Backing Up and Restoring Data*](#)
- [*Moving or Restoring a User Database*](#)
- [*Comparing the Data in Two Directory Servers*](#)

Backing Up and Restoring Data

Administrators should have a comprehensive backup strategy and schedule that comprise of daily, weekly, and monthly backups including incremental and full backups of the directory server data, configuration, and backends. Administrators should also have a backup plan for the underlying filesystem; for example, taking regular snapshots on ZFS systems on Solaris-based machines. This dual purpose approach provides excellent coverage in the event that a server database must be restored for any reason.

The Ping Identity Directory Server provides efficient `backup` and `restore` command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler. The Directory Server can run backups with the server online while processing other requests, so that there is no need to shut down the server or place it in read-only mode prior to starting a backup.

If you back up more than one backend, the `backup` tool creates a subdirectory below a specified backup directory for each backend. If you back up only a single backend, then the backup files will be placed in the specified directory. A single directory can only contain files from one backend, so that you cannot have backup files from multiple different backends in the same backup directory.

When performing a backup, the server records information about the current state of the server and backend, including the server product name, the server version, the backend ID, the set of base DNs for the backend, and the Java class used to implement the backend logic. For JE backends, the backup descriptor also includes information about the Berkeley DB JE version and information about the attribute and VLV indexes that have been defined.

When restoring a backup, the server compares the descriptor obtained from the backup with the current state of the server and backend. If any problems are identified, the server generates warnings or errors. The administrator can choose to ignore the warnings with the `ignoreCompatibilityWarnings` option to the `restore` tool, whereas errors will always cause the restore to fail. For example, when restoring a *newer* backup into an older version of the server, a warning will be generated. When restoring an *older* backup into a new version of the server, no warning will be generated, but because the `config` and `schema` backends require special handling, the server generates an error if the server versions do not match exactly (major, minor, point, and patch version numbers).

The Directory Server's `backup` command and ZFS snapshots provide good coverage in the event a database needs to be restored. For example, the server database would require restoration if the following occurs: 1) the underlying disks have gone bad and the database is inaccessible, 2) the filesystem is operational but the database has become out-of-sync, 3) some application error has occurred data. The Directory Server's `backup` command can help with the first case as it provides a way to store backups on a separate disk from the database itself. ZFS snapshots can help with the last two cases.

To List the Available Backups on the System

- Use the `restore` tool to list the backups in a `backup` directory.


```
$ bin/restore --listBackups --backupDirectory /mybackups
```

```
[13:26:21] The console logging output is also available in '/ds/PingDirectory/logs/tools/restore.log'
```

```
Backup ID:          20120212191715Z
Backup Date:        12/Feb/2012:13:17:19 -0600
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:   false
Dependent Upon:    none
Backup ID:          20120212192411Z
Backup Date:        12/Feb/2012:13:24:16 -0600
Is Incremental:     true
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:   false
Dependent Upon:    20120212191715Z
```

To Back Up All Backends

- Use `backup` to save the all of the server's backends. The optional `--compress` option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

```
$ bin/backup --backUpAll --compress --backupDirectory /path/to/backup
```

To Back Up a Single Backend

- Go to the server root directory, and use the `backup` tool to save the single backend, `userRoot`.

```
$ bin/backup --backendID userRoot --compress --backupDirectory /path/to/backup
```

To Perform an Offline Restore

- Use the `restore` command to restore the `userRoot` backend. Only a single backend can be restored at a time. The Directory Server must be shut down before performing an offline restore.

```
$ bin/restore --backupDirectory /path/to/backup/userRoot
```



Note: The server root directory should never be restored from a file system backup or snapshot.

To Assign an ID to a Backup

- Go to the server root directory, and use the `backup` tool to save the single backend, `userRoot`. The following command assigns the backup ID "weekly" to the `userRoot` backup. The backup file appears under `backups/userRoot` directory as `userRoot-backup-weekly`.

```
$ bin/backup --backupDirectory /path/to/backups/userRoot \  
--backendID userRoot --backupID weekly
```

To Run an Incremental Backup on All Backends

The Directory Server provides support for incremental backups, which backs up only those items that have changed since the last backup (whether full or incremental) on the system, or since a specified earlier backup. Incremental backups must be placed in the same backup directory as the full backup on which they are based.

Not all backends support incremental backups. If a backend does not support incremental backups, use of the `--incremental` option will have no effect, and a full backup will be taken.

- The following command runs an incremental backup on all backends based on the most recent backup:

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup on a Single Backend

- Go to the server root directory, and use `backup` to save the single backend, `userRoot`.

```
$ bin/backup --backendID userRoot --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup based on a Specific Prior Backup

- You can run an incremental backup based on a specific prior backup that is not the most current version on the system. To get the backup ID, use the `restore --listBackups` command (see below).

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup \  
--incrementalBaseID backup-ID
```

To Restore an Incremental Backup

The process for restoring an incremental backup is exactly the same as the process for restoring a full backup for both the online and offline restore types. The `restore tool` will automatically ensure that the full backup and any intermediate incremental backups are restored first before restoring the final incremental backup. The tool will not restore any files in older backups that are no longer present in the final data set.

To Schedule an Online Backup

You can schedule a backup to run as a Task by specifying the timestamp with the `--task` and `--start` options. The option is expressed in "YYYYMMDDhhmmss" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

For online (remote) backups, the backup operation can be conducted while the Ping Identity Directory Server is online if you provide information about how to connect and to authenticate to the target Directory Server.

- You can schedule the backup to occur at a specific time using the Task-based `--start YYYYMMDDhhmmss` option. To specify a time in the UTC time zone format, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the time zone configured on the server.

```
$ bin/backup --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com
```

```
Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

To Schedule an Online Restore

By providing connection and authentication information (and an optional start time), the restore can be performed via the Tasks subsystem while the server is online. The Tasks subsystem allows you to schedule certain operations, such as `import-ldif`, `backup`, `restore`, `start-ds`, and `stop-ds`. You can schedule a restore to run as a Task by specifying the timestamp with the `--task` and `--start` options. The option is expressed in “YYYYMMDDhhmmss” format. If the option has a value of “0” then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

- The backend that is being restored will be unavailable while the restore is in progress. To specify a time in the UTC time zone, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the configured time zone on the server.

```
$ bin/restore --task --start 20111025010000 \
  --backupDirectory /path/to/backup/userRoot \
  --completionNotify admin@example.com --errorNotify admin@example.com
```

To Encrypt a Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, `userRoot` and encrypt it with the `--encrypt` option.

```
$ bin/backup --encrypt --backendID userRoot --compress --backupDirectory /path/to/
backup
```

To Sign a Hash of the Backup

- Go to the server root directory, and use the `backup` tool to backup up the single backend, `userRoot`. Use the `-signHash` option to generate a hash of the backup contents and digitally sign the hash of the backup contents. If you want to generate only a hash of the backup contents, run `backup` with the `--hash` option.

```
$ bin/backup --signHash --backupDirectory backups/userRoot --backendID userRoot \
  --backupDirectory /path/to/backup
```

To Restore a Backup

- Go to the server root directory, and use the `restore` tool to restore a backup. The `backup` tool uses a descriptor file to access property information used for the backup, indicating if the backup was compressed, signed and/or encrypted.

```
$ bin/restore --backupDirectory /path/to/backup
```

Moving or Restoring a User Database

Part of any disaster recovery involves the restoration of the user database from one server to another. You should have a well-defined backup plan that takes into account whether or not your data is replicated among a set of servers. The plan is the best insurance against significant downtime or data loss in the event of unrecoverable database issue.

Keep in mind the following general points about database recovery:

- **Regular Backup from Local Replicated Directory Server.** Take a backup from a local replicated directory server and restore to the failed server. This will be more recent than any other backup you have.
- **Restore the Most Recent Backup.** Restore the most recent backup from a local server. In some cases, this may be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it will take longer for replication to play back changes.
- **Contact Support.** If all else fails, contact your authorized support provider and they can work with you (and possibly in cooperation with the Oracle Berkeley DB JE engineers) to try a low-level recovery, including tools that attempt to salvage whatever data they can obtain from the database.

Comparing the Data in Two Directory Servers

The Ping Identity Directory Server provides an `ldap-diff` tool to compare the data on two LDAP servers to determine any differences that they may contain. The differences are identified by first issuing a subtree search on both servers under the base DN using the default search filter (`objectclass=*`) to retrieve the DNs of all entries in each server. When the tool finds an entry that is on both servers, it retrieves the entry from each server and compares all of its attributes. The tool writes any differences it finds to an LDIF file in a format that could be used to modify the content of the source server, so that it matches the content of the target server. Any non-synchronized entries can be compared again for a configurable number of times with an optional pause between each attempt to account for replication delays.

You can control the specific entries to be compared with the `--searchFilter` option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. You can also exclude specific attributes by prepending a `^` character

to the attribute. (On Windows operating systems, excluded attributes must be quoted, for example, "^attrToExclude".) The @objectClassName notation can be used to compare only attributes that are defined for a given objectclass.

The `ldap-diff` tool can be used on servers actively being modified by checking differing entries multiple times without reporting false positives due to replication delays. By default, it will re-check each entry twice, pausing two seconds between checks. These settings can be configured with the `--numPasses` and `--secondsBetweenPass` options. If the utility cannot make a clean comparison on an entry, it will list any exceptions in comments in the output file.

The Directory Server user specified for performing the searches must be privileged enough to see all of the entries being compared and to issue a long-running, unindexed search. For the Directory Server, the out-of-the-box `cn=Directory Manager` user has these privileges, but you can assign the necessary privileges by setting the following attributes in the user entry:

```
ds-cfg-default-root-privilege-name: unindexed-search
ds-cfg-default-root-privilege-name: bypass-acl
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-lookthrough-limit: 0
```

The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For Directory Servers that contain hundreds of millions of entries, the tool might require a few gigabytes of memory. If the progress of the tool slows dramatically, it might be running low on memory. The memory used by the `ldap-diff` tool can be customized by editing the `ldap-diff.java-args` setting in the `config/java.properties` file and running the `dsjavaproperties` command.

If you do not want to use a subtree search filter, you can use an input file of DNs for the source, target, or both. The format of the file can accept various syntaxes for each DN:

```
dn: cn=this is the first dn
dn: cn=this is the second dn and it is wrapped cn=this is the third dn
# The following DN is base-64 encoded dn::
Y249ZG9uJ3QgeW91IGhhdmUgYmV0dGVyIHROaW5ncyB0byBkbyB0aGFuIHNlZSB3aGF0IHROaXMgc2F5cw==
# There was a blank line above dn: cn=this is the final entry.
```



Caution: Do not manually update the servers when the tool identifies differences between two servers involved in replication. First contact your authorized support provider for explicit confirmation, because manual updates to the servers risk introducing additional replication conflicts.

To Compare Two Directory Servers Using `ldap-diff`

1. Use `ldap-diff` to compare the entries in two Directory Server instances. Ignore the `userpassword` attribute due to the one-way password hash used for the password storage scheme.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
--sourceHost server1.example.com --sourcePort 1389 \
--sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
--targetHost server2.example.com --targetPort 2389 \
--targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
--baseDN dc=example,dc=com --searchFilter "(objectclass=*)" "^userpassword"
```

2. Open the output file in a text editor to view any differences. The file is set up so that you can re-apply the changes without any modification to the file contents. The file shows any deletes, modifies, and then adds from the perspective of the source server as the authoritative source.

```
# This file contains the differences between two LDAP servers.
#
# The format of this file is the LDIF changes needed to bring server
# ldap://server1.example.com:1389 in sync with server
# ldap://server2.example.com:2389.
#
# These differences were computed by first issuing an LDAP search at both
# servers under base DN dc=example,dc=com using search filter (objectclass=*)
# and search scope SUB to first retrieve the DNS of all entries. And then each
# entry was retrieved from each server and attributes: [^userpassword] were
# compared. # # Any entries that were out-of-sync were compared a total of 3 times
# waiting a minimum of 2 seconds between each attempt to account for replication
# delays.
#
# Comparison started at [24/Feb/2010:10:34:20 -0600]
# The following entries were present only on ldap://server2.example.com:2389 and
# need to be deleted. This entry existed only on ldap://server1.example.com:1389
# Note: this entry might be incomplete. It only includes attributes:
# [^userpassword]dn: uid=user.200,ou=People,dc=example,dc=com
# objectClass: person
# objectClass: inetOrgPerson
... (more attributes not shown) ...
# st: DC
dn: uid=user.200,ou=people,dc=example,dc=com
changetype: delete

# The following entries were present on both servers but were out of sync.

dn: uid=user.199,ou=people,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 300 848 9999
-
delete: mobile
mobile: +1 009 471 1808

# The following entries were missing on ldap://server2.example.com:2389 and need
# to be added. This entry existed only on ldap://server2.example.com:2389
# Note: this entry might be incomplete. It only includes attributes:

# [^userpassword]
dn: uid=user.13,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
... (more attributes not shown) ...
# Comparison completed at [24/Feb/2010:10:34:25 -0600]
```

To Compare Configuration Entries Using ldap-diff

- Use `ldap-diff` to compare the configuration entries in two Directory Server instances. The filter searches all configuration entries. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
--sourceHost server1.example.com --sourcePort 1389 \
--sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
--targetHost server2.example.com --targetPort 2389 \
--targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
--baseDN cn=config --searchFilter "(objectclass=*)" "^userpassword"
```

To Compare Entries Using Source and Target DN Files

- Use `ldap-diff` to compare the entries in two Directory Server instances. In the following example, the utility uses a single DN input file for the source and target servers, so that no search filter is used. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \  
--sourceHost server1.example.com --sourcePort 1389 \  
--sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \  
--targetHost server2.example.com --targetPort 2389 \  
--targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \  
--baseDN "dc=example,dc=com" --sourceDNsFile input-file.ldif \  
--targetDNsFile input-file.ldif "^userpassword"
```

To Compare Directory Servers for Missing Entries Only Using `ldap-diff`

- Use `ldap-diff` to compare two Directory Servers and return only those entries that are missing on one of the servers using the `--missingOnly` option, which can significantly reduce the runtime for this utility.

```
$ bin/ldap-diff --outputLDIF difference.ldif \  
--sourceHost server1.example.com --sourcePort 1389 \  
--sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \  
--targetHost server2.example.com --targetPort 2389 \  
--targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \  
--baseDN dc=example,dc=com --searchFilter "(objectclass=*)" "^userpassword" \  
--missingOnly
```


Chapter

10 Working with Indexes

The Ping Identity Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server.

This chapter presents topics related to indexes:

Topics:

- [Overview of Indexes](#)
- [General Tips on Indexes](#)
- [Index Types](#)
- [System Indexes](#)
- [Managing Local DB Indexes](#)
- [Working with JSON Indexes](#)
- [Working with Local DB VLV Indexes](#)
- [Working with Filtered Indexes](#)
- [Tuning Indexes](#)

Overview of Indexes

The Ping Identity Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server. The Directory Server automatically creates index files when you first initialize a base DN or when you use the `dsconfig` tool to create a local DB backend. During modify operations, the Directory Server updates the database index files.

The Ping Identity Directory Server comes with the following types of indexes:

- **Default system indexes** to ensure that the server operates efficiently. Indexes consist of database files that contain index keys mapping to the list of entry IDs.
- **Default Local DB indexes** that are created for each database suffix. Modify the index to meet your system's requirements using the `dsconfig` tool.
- **Local DB VLV indexes** that allow a client to request the server to send search results using the Virtual List View control.
- **Filtered Indexes** that provide the ability to index an attribute but only for entries that match a specified filter based on an equality index. The filtered index can only be used for searches containing that filter. The filtered index can be maintained independently of the equality index for that attribute and even if a normal equality index is not maintained for that attribute.

General Tips on Indexes

Administrators should keep the following tips in mind when working with indexes:

- **Important Critical Indexes.** The Directory Server has several built-in indexes on the Local DB Backend that are critical to internal server processing and should never be removed.

```
aci, ds-entry-unique-id, objectClass
```

- **Built-in Indexes for Efficient Queries.** The Directory Server has built-in indexes on the Local DB Backend that support efficient queries over standard LDAP. These indexes are not required for all deployments.

```
cn, givenName, mail, member, sn, telephoneNumber, uid, uniqueMember
```

If the data in the directory does not include these attributes, then the indexes can be removed. Note however that there is also no cost to having them present either. If the data includes these attributes, but they are not used in queries, then they can be removed, which will reduce the size of the database both on disk and in memory.

- **Online Rebuilds.** Whenever an online index rebuild is in progress, the data in that backend will be available and writable although the index being rebuilt will not be used; therefore, searches which attempt to use that attribute might be unindexed.
- **Index Rebuild Administrative Alert.** The Directory Server generates an administrative alert when the rebuild process begins and ends. It will have a degraded-alert-type of "index-rebuild-in-progress" so that a Directory Proxy Server, such as the Directory Server can avoid using that server while the rebuild is in progress.
- **System Indexes Cannot be Rebuilt.** The contents of the backend must be exported and re-imported in order to rebuild system indexes. See the table below for the list of system indexes.
- **Indexing Certain Attributes.** You should ensure that the following recommendations are used when setting up the indexes.
 - Equality and substring indexes should not be used for attributes that contain binary data.
 - Approximate indexes should be avoided for attributes containing numbers, such as telephone numbers.
- **Unindexed Searches.** Attributes that are not indexed are unindexed and result in longer search times as the database itself has to be searched instead of the database index file. Only users with the `unindexed-search` privilege are allowed to carry out unindexed searches. In general, applications should be prevented from performing unindexed searches, so that searches that are not indexed would be rejected rather than tying up a worker thread. The ways to achieve this include:
 - Make sure that only the absolute minimum set of users have the `unindexed-search` privilege.
 - Make sure that `allow-unindexed-searches` property is set to false in all client connection policies in which unindexed searches should never be necessary.
 - Set a nonzero value for the `maximum-concurrent-unindexed-searches` global configuration property to ensure that if unindexed searches are allowed, only a limited number of them will be active at any given time. Administrators can configure the maximum number of concurrent unindexed searches by setting a property under Global Configuration.

To change the maximum number of concurrent unindexed searches, use the `dsconfig` tool to set a value for the number. A value of "0" (default) represents no limit on the number of concurrent unindexed searches.

```
$ bin/dsconfig set-global-configuration-prop \  
--set maximum-concurrent-unindexed-searches:2
```

- **Index Entry Limit.** The Directory Server specifies an index entry limit property. This property defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (default value is 4000), then you must rebuild the indexes using the `rebuild-index` tool. If an index entry limit value is set for the local DB backend, it overrides the value set for the overall JE backend index entry limit configuration (i.e., 4000).

To change the default index entry limit, use the `dsconfig` tool as seen in the following example:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name cn --set index-entry-limit:5000
```

- **Rebuild Index vs Full Import.** You can expect a limited amount of database growth due to the existence of old data when running `rebuild-index` versus doing a full import of your database.

Index Types

The Ping Identity Directory Server supports several types of indexes to quickly find entries that match search criteria in LDAP operations. The Directory Server uses an attribute's matching rules to normalize its values and uses those values as index keys to a list of matching entry IDs. Entry IDs are integer values that are used to uniquely identify an entry in the backend by means of a set of database index files (`id2entry`, `dn2id`, `dn2uri`, `id2children`, `id2subtree`).

Matching rules are elements defined in the schema that tell the server how to interact with the particular attribute. For example, the `uid` attribute has an equality matching rule defined in the schema, and thus, has an equality index maintained by the Directory Server. The following table describes the index types:

Table 14: Directory Server Index Types

Index Type	Description
Approximate	Used to efficiently locate entries that match the approximate search filter. It is used to identify which entries are approximately equal to a given assertion. Approximate indexes can only be applied to attributes that have a corresponding approximate matching rule.
Equality	Used to efficiently locate entries that match the equality search filter. It is used to identify which entries are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule. An offshoot of the equality index is the filtered index, which uses a defined search filter for a specific attribute. The filtered index can be maintained independently of the equality index for a specific attribute.
Ordering	Used to efficiently locate entries that match the ordering search filter. It is used to identify which entries have a relative order of values for an attribute. Ordering indexes can only be applied to attributes that have a corresponding ordering matching rule.
Presence	Used to efficiently locate entries that match the presence search filter. It is used to identify which entries have at least one value for a specified attribute. There is only one presence index key per attribute.
Substring	Used to efficiently locate entries that match the substring search filter. It is used to identify which entries contain specific substrings to a given assertion. Substring indexes can only be applied to attributes that have a corresponding substring matching rule.

System Indexes

The Ping Identity Directory Server contains a set of system database index files that ensure that the server operates efficiently. These indexes cannot be modified or deleted.

Table 15: System Indexes

Index	Description
dn2id	Allows quick retrieval of DNs. The DN database, or <code>dn2id</code> , has one record for each entry. The key is the normalized entry DN and the value is the entry ID.
id2entry	Allows quick retrieval of entries. The <code>id2entry</code> database contains the LDAP entries. The database key is the entry ID and the value is the entry contents.
referral	Allows quick retrieval of referrals. The referral database called <code>dn2uri</code> contains URIs from referral entries. The key is the DN of the referral entry and the value is that of a labeled URI in the <code>ref</code> attribute for that entry.
id2children	Allows quick retrieval of an entry and its children. The <code>id2children</code> database provides a mapping between an entry's unique identifier and the entry unique identifiers of the corresponding entry's children.
id2subtree	Allows quick retrieval of an entry's subtree. The <code>id2subtree</code> database provides a mapping between an entry's unique identifier and its unique identifiers in its subtree.

To View the System Indexes

Use the `dbtest` command to view the system and user indexes. The index status indicates if it is a trusted index or not. An index is either trusted or untrusted. An untrusted index requires rebuilding.

```
$ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID userRoot
```

Managing Local DB Indexes

You can modify the local DB indexes to meet your system's requirements using the `dsconfig` tool. If you are using the `dsconfig` tool in interactive command-line mode, you can access the **Local DB Index** menu from the Basic object menu.

To View the List of Local DB Indexes

- Use `dsconfig` with the `list-local-db-indexes` option to view the default list of indexes.

```
$ bin/dsconfig list-local-db-indexes --backend-name userRoot
```

```
Local DB Index      : Type      : index-type
-----:-----:-----
aci                 : generic  : presence
cn                  : generic  : equality, substring
ds-entry-unique-id : generic  : equality
givenName           : generic  : equality, substring
mail                : generic  : equality
member              : generic  : equality
objectClass         : generic  : equality
sn                  : generic  : equality, substring
telephoneNumber     : generic  : equality
uid                 : generic  : equality
uniqueMember        : generic  : equality
```

To View a Property for All Local DB Indexes

- Use `dsconfig` with the `--property` option to view a property assigned set for all local DB indexes. Repeat the option for each property that you want to list. In this example, the `prime-index` property specifies if the backend is configured to prime the index at startup.

```
$ bin/dsconfig list-local-db-indexes --property index-entry-limit \  
--property prime-index --backend-name userRoot
```

To View the Configuration Parameters for Local DB Index

- To view the configuration setting of a local DB index, use `dsconfig` with the `get-local-db-index-prop` option and the `--index-name` and `--backend-name` properties. If you want to view the advanced properties, add the `--advanced` option to your command.

```
$ bin/dsconfig get-local-db-index-prop --index-name aci \  
--backend-name userRoot
```

To Modify the Configuration of a Local DB Index

You can easily modify an index using the `dsconfig` tool. Any modification or addition of an index requires the indexes to be rebuilt. In general, an index only needs to be built once after it has been added to the configuration.

If you add an index, then import the data using the `import-ldif` tool, then the index will be automatically rebuilt. If you add an index, then add the data using some other method than `import-ldif`, you must rebuild the index using the `rebuild-index` tool.

1. Use `dsconfig` with the `set-local-db-index-prop` option and the `--index-name` and `--backend-name` properties. In this example, update the `prime-index` property, which loads the index at startup. This command requires the `--advanced` option to access this property.

```
$ bin/dsconfig set-local-db-index-prop --index-name uid \  
--backend-name userRoot --set prime-index:true
```

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-index-prop --index-name uid \  
--backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in [General Tips on Indexes](#).

```
$ bin/stop-ds
```

4. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index uid
```

5. Restart the Directory Server if shutdown.

```
$ bin/start-ds
```

To Create a New Local DB Index

1. To create a new local DB index, use `dsconfig` with the `--create-local-db-index` option and the `--index-name`, `--backend-name`, and `--set index-type: <value>` options.

```
$ bin/dsconfig create-local-db-index \
  --index-name roomNumber --backend-name userRoot \
  --set index-type:equality
```

2. View the index.

```
$ bin/dsconfig get-local-db-index-prop \
  --index-name roomNumber --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in *General Tips on Indexes* on page 148.

```
$ bin/stop-ds
```

4. Rebuild the index using the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the Directory Server.

```
$ bin/start-ds
```

To Delete a Local DB Index

You can delete an index using the `dsconfig` tool. Check that no plug-in applications are using the index before deleting it. When the index is deleted, the corresponding index database will also be deleted. The disk space is reclaimed once the cleaner threads begin.

- Use `dsconfig` with the `delete-local-db-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot
```

Working with JSON Indexes

JSON indexing is similar to general attribute indexing. Where an attribute can be indexed several ways and requires a separate database for each index type, there is only a single database for each JSON field that can be used for different JSON filter types. This database primarily behaves like the database for an equality attribute index. Each database entry key is the normalized form for a value for the target JSON field, and the corresponding database entry value is a list of the entry IDs for all entries in which the associated attribute type has a JSON object with that value for the target field. The database is configured with a comparator (based

on the data type for the target field) that enables iterating through values in a logical order to facilitate inequality and subInitial searches.

JSON indexes are automatically created when the JSON Field constraint indicates that a JSON field should be indexed. Indexes can be viewed with the following command:

```
$ bin/dbtest list-index-status \  
--backendID userRoot \  
--baseDN dc=example,dc=com
```

The JSON object filter types that can be enhanced through the use of JSON indexes include:

- `equals`. Identifies entries that have a specific value for the target field. This filter type only requires retrieving a single index key. However, depending on the nature of the search filter, the ID list may contain references to entries that don't actually match the filter (such as if the field is a string, and the filter is configured to use case-sensitive matching).
- `equalsAny`. Identifies entries that have any of a specified set of values for the target field. This filter type only requires retrieving the index keys that correspond to the target values in the filter and merging their ID lists.
- `greaterThan/lessThan`. Identifies entries that have at least one value for the target field that is greater or less than (or possibly equal to) a specified value. This index is similar in use to the `containsField` index, except that it only needs to iterate through a subset of the keys. A filter can contain both `greaterThan` and `lessThan` filters to represent a bounded range.
- `substring`. Identifies entries that have a string value for the target field that matches a given `substring`. The index can only be used for substring filters that include a `subInitial` component. In this case, the server iterates through all of the index keys that match the `startsWith` component, and manually compares values against the remainder of the `substring` assertion.

JSON indexing is available in local database backends backed by Berkeley DB Java Edition. This includes the following:

- Add, delete, modify, and modify DN operations that make changes to JSON objects stored in the server.
- LDIF imports that include JSON objects, including updates to the cache size estimates for the JSON indexes.
- The `rebuild-index` tool and corresponding backend code to make it possible to generate and rebuild indexes for JSON data. It must be possible to build all JSON indexes for all or a specified subset of fields associated with a given attribute type. The `verify-index` tool should also work with JSON indexes to make it possible to check their validity.
- Matching entry count control and `debugsearchindex` return attribute provide information about relevant JSON index usage.
- Support for monitoring index content and usage.



Note: Exploded indexes and the entry balancing global index do not support JSON objects.

Working with Local DB VLV Indexes

Local DB VLV indexes allow a client to request a subset of results from a sorted list that match a specific search base, scope, and filter. The client can navigate through the list by passing a context back to the server with the virtual list view control. The Local DB VLV index can be used only when the client request contains the virtual list view (VLV) control and the client has been authorized with an ACI with a `targetcontrol` of **2.16.840.1.113730.3.4.9**.



Note: A client request, which includes a virtual list view control, can be successfully processed without a matching Local DB VLV index if the search is completely indexed. This is not an efficient means of using VLV, since the server has to retrieve each entry twice.

To View the List of Local DB VLV Indexes

- Use `dsconfig` with the `list-local-db-indexes` option to view the default list of indexes. In the example, no VLV indexes are defined.

```
$ bin/dsconfig list-local-db-vlv-indexes --backend-name userRoot
```

To Create a New Local DB VLV Index

1. Use `dsconfig` with the `create-local-db-vlv-index` option and the `--index-name`, `--backend-name`, and `--set index-type:(propertyValue)` options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-vlv-index \
  --index-name givenName --backend-name userRoot --set base-dn:dc=example,dc=com \
  --set scope:whole-subtree --set filter:"(objectclass=*)" \
  --set sort-order:givenName
```

2. Rebuild the index using the `rebuild-index` tool. You must add the “vlv.” prefix to the index name to rebuild the VLV index. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Modify a VLV Index’s Configuration

1. Use `dsconfig` with the `set-local-db-vlv-index-prop` option and the `--index-name` and `--backend-name` properties. In this example, update the `base-dn` property.

```
$ bin/dsconfig set-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --set base-dn:ou=People,dc=example,dc=com
```

2. Rebuild the index using the `rebuild-index` tool. You must add the prefix "vlv." to the index name. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

To Delete a VLV Index

You can delete a VLV index using the `dsconfig` tool. Check that the index is not being used in any plug-in applications before deleting it.

1. Use `dsconfig` with the `delete-local-db-vlv-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-vlv-index --index-name givenName \  
--backend-name userRoot
```

2. Verify the deletion by trying to view the vlv index.

```
$ bin/dsconfig get-local-db-vlv-index-prop --index-name givenName \  
--backend-name userRoot
```

Working with Filtered Indexes

The Ping Identity Directory Server filtered index is useful when client search requests consisting of a compound `&`-filter with individual components matching a large number of entries, potentially greater than the index entry limit, have an intersection of a relatively small number of entries.

For example, assume a database contains several thousand company profiles and each company profile is represented by many entries. The `"(objectClass=company)"` filter matches a small set of entries per company, and therefore may exceed the index entry limit since there are many companies. Also assume that the `"(companyDomain=example.com)"` filter matches many of the entries for the company with domain `example.com` and can also result in an unindexed search. The more narrow filter `"(&(objectClass=company)(companyDomain=example.com))"` also results in an unindexed search but only matches a small number of entries. The filtered index makes it possible to index this compound filter by defining an equality index on the `companyDomain` attribute with a static filter of `"(objectClass=company)"` in the `equality-index-filter` property of the index.

Filtered indexing is primarily useful for cases in which clients frequently issue searches with AND filters that meet the following criteria:

- The AND filter itself matches a relatively small number of entries, but each of the individual components may match a very large number of entries.
- The filter has a dynamic component that does change, and that dynamic component always uses the same attribute.
- The filter has a static component that doesn't change.

The filtered index can be maintained independently from the equality filter for that attribute. Further, the filtered index will be used only for searches containing the equality component with the associated attribute type ANDed with this filter. When configuring a filtered index, be aware of the `equality-index-filter` and `maintain-equality-index-without-filter` properties of the index.

Once configured and built with the `rebuild-index` tool or `import-ldif`, searches with filters based on the above example will be processed with the index:

```
(&(objectClass=company)(companyDomain=example.com))
(&(objectClass=company)(|(companyDomain=example.com)(companyDomain=example.org)))
(&(companyDomain=example.com)(objectClass=company))
(&(companyDomain=example.com)(&(objectClass=company)))
(&(companyDomain=example.com)(objectClass=company)(something=else))
(&(companyDomain=example.com)(&(objectClass=company)(something=else)))
(|(&(objectClass=company)(companyDomain=example.com))(&(objectClass=company)
(companyDomain=example.org)))
```

When configuring a filtered index, define the following properties:

Table 16: Filtered Index Properties

Filtered Index Properties	Description
<code>equality-index-filter</code>	Specifies a search filter that may be used in conjunction with an equality component for the associated attribute type. If an equality index filter is defined, then an additional equality index will be maintained for the associated attribute, but only for entries that match the provided filter. Further, the index will be used only for searches containing an equality component with the associated attribute type ANDed with this filter.
<code>maintain-equality-index-without-filter</code>	Specifies whether to maintain a separate equality index for the associated attribute without any filter, in addition to maintaining an index for each equality index filter that is defined. If this is false, then the attribute will not be indexed for equality by itself but only in conjunction with the defined equality index filters.

To Create a Filtered Index

1. Use the `dsconfig` tool to create a filtered index. The following command creates an equality index on the `companyDomain` attribute and maintains an index for the equality filter defined `"(objectclass=company)"`. After you have created the index, you must rebuild the indexes.

```
$ bin/dsconfig create-local-db-index --backend-name "userRoot" \
--index-name companyDomain --set maintain-equality-index-without-filter:true \
--set index-type:equality --set equality-index-filter:"(objectclass=company)"
```

2. Stop the Directory Server using `bin/stop-ds`.
3. Run the `rebuild-index` tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index companyDomain
```

4. Start the Directory Server using `bin/start-ds`.

Tuning Indexes

The Ping Identity Directory Server provides several tools to help you optimize your indexes and improve the overall read and write performance for your system. The server now supports

an optional expanded index database using an *exploded* format to process high write load operations. To view the current state of the server's indexes and make adjustments to the index databases, the Directory Server automatically generates an Index Summary Statistics Table after each LDIF import or index rebuild. The `dbtest` tool also includes an Index Histogram to determine the key datasize for the indexes. This section provides descriptions of each of these tools.

About the Exploded Index Format

The Directory Server has two index properties on the Backend configuration object that can be used to optimize the server's performance: `index-entry-limit` and `exploded-index-entry-threshold`. The `index-entry-limit` specifies the maximum size of entries kept in an index record, before the server stops maintaining that record, begins scanning the whole database, and runs an expensive unindexed search. Typically, if any index keys have already reached this limit, indexes must be rebuilt before they can be allowed to use a new limit.

The `exploded-index-entry-threshold` property specifies the maximum number of entries that are allowed to match a given index key before that particular index key is stored in a separate database in an expanded (i.e., "exploded") format. An `exploded-index-entry-threshold` value of 0, or any value greater or equal to the `index-entry-limit`, means that the expanded index database will not be used. All keys whose entry count is less than `exploded-index-entry-threshold` continue to be stored in one database in a consolidated format, such that changes to the key require *rewriting* all the entry IDs matching the key. All keys whose entry count is greater than `exploded-index-entry-threshold` and less than the `index-entry-limit` are stored in a *separate* database in an exploded format, such that changes to the key require writing only to the updated entry ID.

For example, as shown in the figure below, the equality index for the `sn` attribute is stored in consolidated format, listing each entry ID for all entries that contain the `sn=smith` attribute. If we enable the `exploded-index-entry-threshold` property by setting its value to 1000 and keep the `index-entry-limit` property to its default limit of 4000, then the expanded index database will be implemented by the server. Any key with an entry count less than 1000 continues to be stored in consolidated format. If a key has an entry count greater than 1000, it will be stored in a separate database where each key is stored with its entry ID individually. The consolidated format is very efficient for read operations because the server can retrieve a row of entry IDs at once, while the exploded format is far more efficient for write operations.

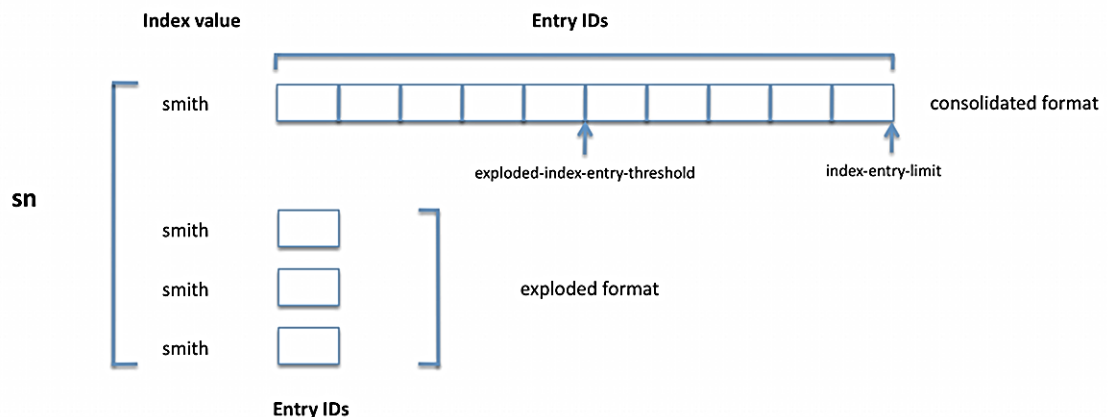


Figure 3: Consolidated and Exploded Index Formats

Note that a large value for the `exploded-index-entry-threshold` can have a big impact on write performance and database growth on disk, since each change to an index key stored in the consolidated format requires that IDs for all entries matching the key be rewritten to the database. For example if the `exploded-index-entry-threshold` is set to 100,000, and the number of entries matching a given key is 50,000, then changes to that key will require writing 200KB to the database since each entry ID is 4 bytes. Add and delete operations are especially impacted since they must update all indexes for an entry. If many indexes have a large `exploded-index-entry-threshold`, this can lead to very low throughput and can cause the database to temporarily grow very large on disk. It is recommended to set this value at or below 10,000, and never set it above 50,000. If any index keys have already reached this threshold, indexes need to be rebuilt before they are allowed to use the new threshold. This is also true for the `index-entry-limit` property: setting a large limit (greater than 10,000) could have a big impact on write performance and database growth on disk.

Monitoring Index Entry Limits

Index keys that have reached their limit require indexes to be rebuilt before they can be allowed to use a new limit. There are several ways to monitor index limits to avoid a potentially costly rebuild. There are cases in which it is acceptable for index keys to exceed the index entry limit. For example, the `objectClass` attribute type should be indexed for equality because the server needs to use it to find all group entries when bringing a backend online, and also because applications frequently need to find entries of a specific type. However, it doesn't make sense for the "top" `objectClass` key to be indexed, because it appears in every entry in the server.

Choose an index entry limit value that is high enough to ensure that all of the right keys are indexed, but keys that occur too frequently are not. The `verify-index` tool `--listKeysNearestIndexEntryLimit` argument lists a specified number of keys that are closest to the limit without having exceeded it. The index entry limit should be larger than the number of entries matching the largest key to remain indexed, with enough overhead to account for future growth. Use this command regularly to determine if the index entry limit needs to be adjusted.

The `verify-index` tool also provides the `--listKeysExceedingIndexEntryLimit` argument to list all keys for which the value has exceeded the index entry limit and the number of entries in which they appear. If there are keys for which the limit has already exceeded but that need to be maintained, adjust the index entry limit to be higher than the number of entries that contain that key (with additional room for future growth) and run the `rebuild-index` tool (or export to LDIF and re-import).

The server provides other ways to determine if index keys have exceeded, or are close to exceeding, the index entry limit. Some of these include:

- When performing an LDIF import, the tool includes an "Index Summary Statistics" section that provides usage information for each index, including the number of keys for which the index entry limit has been exceeded, and also the number of keys for which the number of matching entries falls within a number of predefined buckets (such as 1–4 entries, 5–9 entries, 10–99 entries, and 100–999 entries).
- If, during a search operation, the server accesses one or more index keys whose values have exceeded the index entry limit, the access log message for that operation will include an `indexesWithKeysAccessedExceedingEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add,

delete, modify, and modify DN operations in which the server wrote to, or tried to write to, at least one index key whose value exceeded the index entry limit.

- If, during a search operation, the server accesses one or more index keys whose values have not yet exceeded the index entry limit but are more than 80 percent to reaching that limit, the access log message for that operation will include an `indexesWithKeysAccessedNearEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to at least one index key whose value was within 80 percent of the index entry limit.
- If a search operation requests either includes the `debugsearchindex` attribute, or the matching entry count request control with debugging enabled, the debug information will include any indexes accessed that have exceeded the index entry limit, or that are within 80 percent of the configured index entry limit.
- The monitor entry for each configured index includes attributes that provide information about the number of index keys that have been encountered (since the backend was brought online, or since the index entry limit was changed) in a number of different categories. These monitor attributes include:
 - `ds-index-exceeded-entry-limit-count-since-db-open` — The number of index keys for which the number of matching entries has crossed the index entry limit due to a write operation.
 - `ds-index-unique-keys-near-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries is within 80 percent of the index entry limit.
 - `ds-index-unique-keys-exceeding-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.
 - `ds-index-unique-keys-near-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries is within 80 percent of the index entry limit.
 - `ds-index-unique-keys-exceeding-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.

About the Index Summary Statistics Table

The Directory Server now automatically generates an Index Summary Statistics Table, which can be used to determine the optimal configuration for your system's indexes. This table is only generated when the `rebuild-index` tool is run in offline mode. The table is generated after any LDIF import or an index rebuild, and is written to system out and to `logs/tools/rebuild-index-summary.txt`. The table lists the current index entry limit (set by the `index-entry-limit` property on the local DB configuration), the number of keys whose entry count exceeds this limit if any, and the maximum entry count for any key in the index. The table then displays

a histogram of the number of keys whose entry falls within a range of values. An example of the Index Summary Statistics table is shown below for the `sn.equality` and the `sn.substring` indexes.

The following figure shows that there are seven substrings whose entry counts exceed the `index-entry-limit` of 4000. Six of the substrings are in the 10000-99999 range with the maximum entry count being 13419. By deduction, one more substring must be present in the 1000-9999 range that exceeds the `index-entry-limit` of 4000. These substrings could be expensive for search operations.

```

--- Index Summary Statistics ---
Index          : Limit : >Limit : Max   : 1-9   : 10-99  : 100-999 : 1000-9999 : 10000-99999
-----
dc_example_dc_com_sn.equality : 4000 :       : 2     : 100000 :         :         :         :
dc_example_dc_com_sn.substring : 4000 : 7     : 13419 : 150814 : 7109   : 407    : 36     : 6

```

Figure 4: Example of an Index Summary Statistics Table

About the `dbtest` Index Status Table

The `dbtest` tool has a `list-all --analyze` option that generates the current status of all of the databases on your system, including all index databases. The table shows the type, entry count (i.e., the number of records in the database), index status (TRUSTED to indicate that the indexes are up-to-date, or UNTRUSTED if the index needs rebuilding), the total data size for each key, the average data size for each key and the maximum data size for each key. Note also that any indexes that are in exploded format are listed on this table.

Index Name	Index Type	JE Database Name	Index St
<code>id2children</code>	Index	<code>dc_example_dc_com_id2children</code>	TRUSTED
<code>id2subtree</code>	Index	<code>dc_example_dc_com_id2subtree</code>	TRUSTED
<code>uid.equality</code>	Index	<code>dc_example_dc_com_uid.equality</code>	TRUSTED
<code>aci.presence</code>	Index	<code>dc_example_dc_com_aci.presence</code>	TRUSTED
<code>ds-soft-delete-timestamp.ordering</code>	Index	<code>dc_example_dc_com_ds-soft-delete-timestamp.ordering</code>	TRUSTED
<code>ds-soft-delete-from-dn.equality</code>	Index	<code>dc_example_dc_com_ds-soft-delete-from-dn.equality</code>	TRUSTED
<code>givenName.equality</code>	Index	<code>dc_example_dc_com_givenName.equality</code>	TRUSTED
<code>givenName.substring</code>	Index	<code>dc_example_dc_com_givenName.substring</code>	TRUSTED
<code>objectClass.equality</code>	Index	<code>dc_example_dc_com_objectClass.equality</code>	TRUSTED
<code>member.equality</code>	Index	<code>dc_example_dc_com_member.equality</code>	TRUSTED
<code>uniqueMember.equality</code>	Index	<code>dc_example_dc_com_uniqueMember.equality</code>	TRUSTED
<code>cn.equality</code>	Index	<code>dc_example_dc_com_cn.equality</code>	TRUSTED
<code>cn.substring</code>	Index	<code>dc_example_dc_com_cn.substring</code>	TRUSTED
<code>sn.equality</code>	Index	<code>dc_example_dc_com_sn.equality</code>	TRUSTED
<code>sn.substring</code>	Index	<code>dc_example_dc_com_sn.substring</code>	TRUSTED
<code>telephoneNumber.equality</code>	Index	<code>dc_example_dc_com_telephoneNumber.equality</code>	TRUSTED
<code>mail.equality</code>	Index	<code>dc_example_dc_com_mail.equality</code>	TRUSTED
<code>ds-entry-unique-id.equality</code>	Index	<code>dc_example_dc_com_ds-entry-unique-id.equality</code>	TRUSTED

Figure 5: `dbtest` Output Including Index Databases

Configuring the Index Properties

By default, the `index-entry-limit` and the `exploded-index-entry-threshold` are set to 4000, which means that the latter property is not enabled due to its having an equal value to that of the `index-entry-limit` property. You can enable the exploded index database by changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value using the `dsconfig` tool.

To Configure the Index Properties

Before running the following commands, be aware that you will need to do an index rebuild on the system, which requires a system shutdown, unless the command is run as a task.

1. Run `dsconfig` and set the `index-entry-limit` and the `exploded-index-entry-threshold` to 5000 and 1000, respectively. By default, both values are set to 4000. Changing the `exploded-index-entry-threshold` property to a value less than the `index-entry-limit` and to a non-zero value enables the exploded index database. Remember to include the `bind` parameters for your system. Once you enter the command, you must confirm that you want to apply the changes to your system.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set index-entry-limit:5000 \
  --set exploded-index-entry-threshold:1000
```

One or more configuration property changes require administrative action or confirmation/notification. Those properties include:

- * `index-entry-limit`: If any index keys have already reached this limit, indexes must be rebuilt before they will be allowed to use the new limit. Setting a large limit (greater than 10,000) could have a big impact on write performance and database growth on disk.
- * `exploded-index-entry-threshold`: If any index keys have already reached this threshold, indexes need to be rebuilt before they are allowed to use the new threshold. Setting a large limit (greater than 10,000) could have a big impact on write performance and database growth on disk.

Continue? Choose 'no' to return to the previous step (yes / no) [yes]: yes

2. Stop the server.

```
$ bin/stop-ds
```

3. Rebuild the index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com \
  --index cn --index givenName --index objectClass \
  --index sn --maxThreads 10
```

4. View the Index Summary Statistics table, which is automatically displayed to system out after running the `rebuild-index` command. You can also access the table at `logs/tools/rebuild-index-summary.txt`. Repeat the last three steps to make more adjustments to your indexes.

```
--- Index Summary Statistics ---
Index : Limit : >Limit : Max : 1-9 : 10-99 : 100-999 : 1000-9999 : 10000-99999 : 100000-999999
-----:-----:-----:-----:-----:-----:-----:-----:-----:-----
dc_example_dc_com_cn.equality : 5000 : : 1 : 100001 : : : : :
dc_example_dc_com_cn.substring : 5000 : 9 : 15401 : 131746 : 22372 : 545 : 39 : 3 :
dc_example_dc_com_givenName.equality : 5000 : : 16 : 3788 : 4817 : : : : :
dc_example_dc_com_givenName.substring : 5000 : 8 : 23809 : 7039 : 12062 : 483 : 42 : 3 :
dc_example_dc_com_objectClass.equality : 5000 : 4 : 100003 : 3 : : : : : : 4
dc_example_dc_com_sn.equality : 5000 : : 8 : 13419 : : : : : :
dc_example_dc_com_sn.substring : 5000 : 9 : 15401 : 33967 : 7055 : 459 : 39 : 3 :
```

The first three columns of numbers provide (1) "Limit" - the index entry limit (or blank if there is no limit), (2) ">Limit" - the number of keys whose entry count exceeds the entry limit, and (3) "Max" - the maximum entry count for any key in the index. The remaining columns provide the number of keys whose entry count falls in the range indicated in the column heading

5. Restart the server.

```
$ bin/start-ds
```


Chapter

11

Managing Entries

The Directory Server is a fully LDAPv3-compliant server that comes with a comprehensive set of LDAP command-line tools to search, add, modify, and delete entries.

This chapter presents the following topics:

Topics:

- [*Searching Entries*](#)
- [*Working with the Matching Entry Count Control*](#)
- [*Adding Entries*](#)
- [*Deleting Entries Using Idapdelete*](#)
- [*Deleting Entries Using Idapmodify*](#)
- [*Modifying Entries Using Idapmodify*](#)
- [*Working with the Parallel-Update Tool*](#)
- [*Working with the Watch-Entry Tool*](#)
- [*Working with LDAP Transactions*](#)

Searching Entries

The Directory Server provides an `ldapsearch` tool to search for entries or attributes within your server. The tool requires the LDAP connection parameters needed to bind to the server, including the `baseDN` option to specify the starting point of the search within the server, and the search scope. The `searchScope` option determines the depth of the search:

- > `base` (search only the entry specified)
- > `one` (search only the children of the entry and not the entry itself)
- > `sub` (search the entry and its descendents)

The `ldapsearch` tool provides basic functionality as specified by the RFC 2254 but provides additional features that takes advantage of the Directory Server's control mechanisms. For more information, run the `ldapsearch --help` function.

To Search the Root DSE

The Root DSE is a special entry that resides at the root of the directory information tree (DIT). The entry holds operational information about the server and its supported controls. Specifically, the root DSE entry provides information about the supported LDAPv3 controls, SASL mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.



Note: The Directory Server provides an option to retrieve the Root DSE's operational attributes and add them to the user attribute map of the generated entry. This feature allows client applications that have difficulty handling operational attributes to access the root DSE using the `show-all-attributes` configuration property. Once this property is set, the associated attribute types are re-created and re-registered as user attributes in the schema (in memory, not on disk). Once you set the property, you can use `ldapsearch` without "+" to view the root DSE.

Use the `dsconfig` tool to set the `show-all-attributes` property to `TRUE`, as follows:

```
$ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true
```

- Use `ldapsearch` to view the root DSE entry on the Directory Server. Be sure you include the "+" to display the operational attributes in the entry.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

To Search All Entries in the Directory Server

- Use `ldapsearch` to search all entries in the Directory Server. The filter `"(objectclass=*)"` matches all entries. If the `--searchScope` option is not specified, the command defaults to a search scope of `sub`:

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)"
```

To Search for an Access Control Instruction

- Use `ldapsearch` to search the `dc=example,dc=com` base DN entry. The filter `"(aci=*)"` matches all `aci` attributes under the base DN, and the `aci` attribute is specified so that only it is returned. The `cn=Directory Manager` bind DN has the privileges to view an ACI.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(aci=*)" aci
```

```
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
   allow (read,search,compare) userdn="ldap:///anyone");
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
   allow (write) userdn="ldap:///self");
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
   allow (all) userdn="ldap:///uid=admin,dc=example,dc=com");
```

To Search for the Schema

- Use `ldapsearch` to search the `cn=schema` entry. The base DN is specified as `cn=schema`, and the filter `"(objectclass=*)"` matches all entries. The command uses a special attribute `"+"` to return all operational attributes:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" "+"
```

To Search for a Single Entry using Base Scope and Base DN

- Use `ldapsearch` to search for a single entry by specifying the base scope and DN:

```
$ bin/ldapsearch --baseDN uid=user.14,ou=People,dc=example,dc=com \
  --searchScope base "(objectclass=*)"
```

To Search for a Single Entry Using the Search Filter

- Search for a single entry by specifying the `sub` scope and a search filter that describes a single entry:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
```

```
--searchScope sub "(uid=user.14)"
```

To Search for All Immediate Children for Restricted Return Values

- Search for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn` and `givenName`. The special attribute "+" returns all operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
--searchScope one '(objectclass=*)' sn givenName "+"
```

To Search for All Children of an Entry in Sorted Order

- Search for all children of the `ou=People,dc=example,dc=com` subtree. The resulting entries are sorted by the server in ascending order by `sn` and then in descending order by `givenName`:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
--searchScope sub --sortOrder sn,-givenName '(objectclass=*)'
```

To Limit the Number of Returned Search Entries and Search Time

- Search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree by specifying a compound filter. No more than 200 entries will be returned and the server will spend no more than 5 seconds processing the request. Returned attributes are restricted to a few operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
--searchScope sub --sizeLimit 200 --timeLimit 5 \
"(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id entryUUID
```

To Get Information about How Indexes are used in a Search Operation

The Ping Identity Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Information about the can be obtained about the server's use of indexes in the course of processing a search operation. The following are two basic ways to accomplish this.

Issue a search request with the desired base DN, scope, and filter, and request that the server return the special `debugsearchindex` attribute. Users will need to be granted access to the `debugsearchindex` operational attribute and the `cn=debugsearch` portion of the DIT with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
--add "global-aci:(targetattr=\"debugsearchindex\")(target=\"ldap:///cn=debugsearch\")
(version 3.0; acl \"Allow members of the Index Debugging Users group
to request the debugsearchindex operational attribute \"; allow
(read,search,compare) groupdn=\"ldap:///cn=Index Debugging
Users,ou=Groups,dc=example,dc=com\");"
```

To issue a search request for the server return the special `debugsearchindex` attribute, use the `ldapsearch` command such as:

```
$ bin/ldapsearch --hostname ds.example.com \
--port 389 --bindDN uid=admin,dc=example,dc=com \
--bindPassword password \
--baseDN dc=example,dc=com \
--searchScope sub "(&(givenName=John)(sn=Doe))" debugsearchindex
dn: cn=debugsearch
debugsearchindex: 0.040 ms - Beginning index processing for search
request with base DN 'dc=example,dc=com', scope wholeSubtree,
and filter (&(givenName=John)(sn=Doe)).
debugsearchindex: 0.067 ms - Unable to optimize the AND filter
beyond what the client already provided.
debugsearchindex: 0.834 ms - Candidate set obtained for single-key
filter (givenName=John) from index dc_example_dc_com_givenName.equality.
Candidate set: CandidateSet(isDefined=true, isExploded=false,
isResolved=true, size=2, originalFilter=(givenName=John),
remainingFilter=null, matchingEntryCountType=UNEXAMINED_COUNT)
debugsearchindex: 0.030 ms - Final candidate set for filter (givenName=John)
obtained from an unexploded index key in dc_example_dc_com_givenName.equality.
Since the scope of the search includes the entire entry container, there is
no need to attempt to further pare down the results based on the search scope.
Candidate set: CandidateSet(isDefined=true, isExploded=false, isResolved=true,
size=2, originalFilter=(givenName=John), remainingFilter=null,
matchingEntryCountType=UNEXAMINED_COUNT)
debugsearchindex: 0.020 ms - Short-circuiting index processing for AND filter
(&(givenName=John)(sn=Doe)) after evaluating single-key component
(givenName=John) because the current ID set size of 2 is within the
short-circuit threshold of 5.
debugsearchindex: 0.030 ms - Obtained a candidate set of size 2 for AND filter
(&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe). Even though
there is still more of the filter to evaluate, the current candidate set is
within the short-circuit threshold of 5, so no additional index processing will
be performed to try to pare down the results based on the remaining filter or the
search scope. Candidate set: CandidateSet(isDefined=true, isExploded=false,
isResolved=true, size=2, originalFilter=(&(givenName=John)(sn=Doe)),
remainingFilter=(sn=Doe), matchingEntryCountType=UPPER_BOUND)
debugsearchindex: 0.016 ms - Completed all index processing. Candidate set:
CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
matchingEntryCountType=UPPER_BOUND)
```

The second way would be to issue a search request with the desired base DN, scope, and filter, and include the matching entry count request control with the debug option set to true. To do this with `ldapsearch`, use a command such as:

```
$ bin/ldapsearch --hostname ds.example.com --port 389 \
--bindDN uid=admin,dc=example,dc=com --bindPassword password \
--baseDN dc=example,dc=com \
--searchScope sub --matchingEntryCountControl examineCount=0:debug "(&(givenName=John)
(sn=Doe))"
Upper Bound on Matching Entry Count: 2
Matching Entry Count Debug Messages:
* naw-desktop:1389 - 0.104 ms - Beginning index processing for search request with
base DN 'dc=example,dc=com', scope wholeSubtree, and filter (&(givenName=John)
(sn=Doe)).
* naw-desktop:1389 - 0.105 ms - Unable to optimize the AND filter beyond what the
client already provided.
* naw-desktop:1389 - 0.614 ms - Candidate set obtained for single-key filter
(givenName=John) from index
dc_example_dc_com_givenName.equality. Candidate set:
CandidateSet(isDefined=true, isExploded=false, isResolved=true,
size=2, originalFilter=(givenName=John), remainingFilter=null,
matchingEntryCountType=UNEXAMINED_COUNT)
* naw-desktop:1389 - 0.090 ms - Final candidate set for filter
(givenName=John) obtained from an unexploded index key in
dc_example_dc_com_givenName.equality. Since the scope of the search
includes the entire entry container, there is no need
to attempt to further pare down the results based on the search scope.
Candidate set: CandidateSet(isDefined=true, isExploded=false, isResolved=true,
size=2, originalFilter=(givenName=John), remainingFilter=null,
matchingEntryCountType=UNEXAMINED_COUNT)
* naw-desktop:1389 - 0.045 ms - Short-circuiting index processing for AND filter
(&(givenName=John)(sn=Doe)) after evaluating single-key component
```

```
(givenName=John) because the current ID set size of 2 is within the short-circuit
threshold of 5.
* naw-desktop:1389 - 0.111 ms - Obtained a candidate set of size 2 for AND filter
(&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe). Even though there is
still more of the filter to evaluate, the current candidate set is within
the short-circuit threshold of 5, so no additional index processing will be
performed
to try to pare down the results based on the remaining filter or the search scope.
Candidate set: CandidateSet(isDefined=true, isExploded=false, isResolved=true,
size=2,
originalFilter=&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
matchingEntryCountType=UPPER_BOUND)
* naw-desktop:1389 - 0.040 ms - Completed all index processing. Candidate set:
CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
originalFilter=&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
matchingEntryCountType=UPPER_BOUND)
* naw-desktop:1389 - The search is partially indexed (candidatesAreInScope=true,
unindexedFilterPortion=(sn=Doe))
* naw-desktop:1389 - Constructing an UPPER_BOUND response with a count of 2
```

Working with the Matching Entry Count Control

The `ldapsearch` command can be used with the `--matchingEntryCountControl` option to determine the count of entries that match a search filter. Users will need to be granted access to this control by its OID `1.3.6.1.4.1.30221.2.5.36` with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
--add "global-aci:(targetcontrol=\"1.3.6.1.4.1.30221.2.5.36\")
(version 3.0; acl \"Allow members of the Index Debugging Users group to
use the matching entry count request control \"; allow (read)
groupdn=\"ldap:///cn=Index Debugging Users,ou=Groups,dc=example,dc=com\");"
```

An `examineCount` control can be used for searches that are at least partially indexed, in order to determine whether to return an examined count, an unexamined count, or an upper bound count. The factors that determine what is returned are:

- A search is fully indexed if indexes can be used to identify the entry IDs for all entries that match the filter without ambiguity. Indexes can also be used to make sure that all of those candidates are within the scope of the search.
- A search is partially indexed if indexes can be used to identify the entry IDs for all entries that match the search criteria, but the candidate list may potentially also include entries that either don't match the filter or are outside the scope of the search.
- A search is unindexed if it is not possible to retrieve a candidate list based on either the filter or the search scope.
- An unexamined count is a count of the exact number of entries that match the search criteria, only through the use of index processing.
- An examined count is the same as an unexamined count, except that all of the candidate entries are examined to determine whether they would have been returned to the client. An examined count may be less than an unexamined count if the set of matching entries includes those that would be removed by access control evaluation, or special entries like LDAP sub-entries, replication conflict entries, or soft-deleted entries.
- An upper bound count is the maximum number of entries that match the criteria, but indicates that the server could not determine exactly how many matching entries there were without examining each candidate, which it did not do.

- If a search is fully indexed, the result is an examined count or an unexamined count. If `alwaysExamine` is true and `examineCount` is greater than or equal to the number of candidates, the result is an examined count. If `alwaysExamine` is false, or if the number of candidates exceeds `examineCount`, the result is an unexamined count.
- If a search is partially indexed, the result is either an examined count or an upper bound count. The `alwaysExamine` flag isn't relevant in this case. If `examineCount` is greater than or equal to the number of candidates, the result is an examined count. If not, the result is an upper bound count.
- If a search is unindexed, the result is either an examined count or an unknown count. If `allowUnindexed` is true, the unindexed search is processed, which can be very expensive. Instead of getting the matching entries back, the examined count is returned. If `allowUnindexed` is false, an unknown count is returned. If `allowUnindexed` is true, the requester needs to have the `unindexed-search` privilege to get the exact count.

The following is a sample `ldapsearch` with the `--matchingEntryCountControl` option:

```
$./ldapsearch \
--bindDN "cn=directory manager" \
--bindPassword password \
--baseDN dc=example,dc=com \
--matchingEntryCountControl examineCount=100:alwaysExamine:allowUnindexed:debug \
"(objectclass=*)"
```

Adding Entries

Depending on the number of entries that you want to add to your Directory Server, you can use the `ldapmodify` tool for small additions. The `ldapmodify` tool provides two methods for adding a single entry: using a LDIF file or from the command line. The attributes must conform to your schema and contain the required object classes.

Adding requests with the `ignore-no-user-modification` control enable a client to include attributes that are not normally allowed from external sources. For example, the `userPassword` attribute is a user-modifiable attribute. An add request with the `ignore-no-user-modification` control allows a one-time exception to the password policy, even if the requesting client does not have the `bypass-pw-policy` privilege. This exception enables specifying pre-encoded passwords.



Note: When adding entries, a server-generated entryUUID value can be added in the RDN. An LDAP client written with the commercial edition of the LDAP SDK for Java can use the `NameWithEntryUUIDRequestControl` to achieve this, or the `ldapmodify` and `parallel-update` tools can be used. See each tool's online help.

To Add an Entry Using an LDIF File

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user.ldif`. For the `userPassword`

attribute, enter the cleartext password. The Directory Server encrypts the password and stores its encrypted value in the server. Make sure that the LDIF file has limited read permissions for only authorized administrators.

```
dn: uid=user.2000,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Toby Hall$73600 Mash Street$Cincinnati, OH 50563 postalCode: 50563
description: This is the description for Toby Hall.
uid: user.2000
userPassword: wordsmith employeeNumber: 2000
initials: TBH
givenName: Toby
pager: +1 596 232 3321
mobile: +1 039 311 9878
cn: Toby Hall
sn: Hall
telephoneNumber: +1 097 678 9688
street: 73600 Mash Street
homePhone: +1 214 233 8484
l: Cincinnati
mail: user.2000@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. If the command is successful, you will see generated success messages with the `"#"` symbol.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

```
# Processing ADD request for uid=user.2000,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2000,ou=People,dc=example,dc=com
```

To Add an Entry Using the Changetype LDIF Directive

RFC 2849 specifies LDIF directives that can be used within your LDIF files. The most commonly used directive is `changetype`, which follows the `dn:` directive and defines the operation on the entry. The main advantage of using this method in an LDIF file is that you can combine adds and modifies in one file.

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user2.ldif`. Note the use of the `changetype` directive in the second line.

```
dn: uid=user.2001,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Seely Dorm$100 Apple Street$Cincinnati, OH 50563
postalCode: 50563
description: This is the description for Seely Dorm.
uid: user.2001
userPassword: pleasantry
employeeNumber: 2001
initials: SPD
givenName: Seely pager: +1 596 665 3344
mobile: +1 039 686 4949
cn: Seely Dorm
sn: Dorm
telephoneNumber: +1 097 257 7542
street: 100 Apple Street
homePhone: +1 214 521 4883
```



```
l: Cincinnati
mail: user.2001@maildomain.net
st: OH
```

2. Use the `ldapmodify` tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. In this example, you do not need to use the `--defaultAdd` or its shortform `-a` option with the command.

```
$ bin/ldapmodify --filename add-user2.ldif
```

To Add Multiple Entries in a Single File

You can have multiple entries in your LDIF file by simply separating each DN and its entry with a blank line from the next entry.

1. Open a text editor and create a couple of entries that conform to your schema. For example, add the following entries in the file and save the file as `add-user3.ldif`. Separate each entry with a blank line.

```
dn: uid=user.2003,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass:
organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...

dn: uid=user.2004,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...
```

2. Use the `ldapmodify` tool to add the entries specified in the LDIF file. In this example, we use the short form arguments for the `ldapmodify` tool.

The `-h` option specifies the host name, the `-p` option specifies the LDAP listener port, `-D` specifies the bind DN, `-w` specifies the bind DN password, `-a` specifies that entries that omit a changetype will be treated as add operations, and `-f` specifies the path to the input file. If the operation is successful, you will see commented messages (those beginning with "#") for each addition.

```
$ bin/ldapmodify -h server.example.com -p 389 \
-D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif

# Processing ADD request for uid=user.2003,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2003,ou=People,dc=example,dc=com
# Processing ADD request for uid=user.2004,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2004,ou=People,dc=example,dc=com
```

Deleting Entries Using `ldapdelete`

You can delete an entry using the `ldapdelete` tool. You should ensure that there are no child entries below the entry as that could create an orphaned entry. Also, make sure that you have properly backed up your system prior to removing any entries.

To Delete an Entry Using `ldapdelete`

Use `ldapdelete` to delete an entry. The following example deletes the `uid=user.14` entry.

```
$ bin/ldapdelete uid=user.14,ou=People,dc=example,dc=com
```

To Delete Multiple Entries Using an LDIF File

1. You can generate a file of DNs that you would like to delete from the Directory Server.

The following command searches for all entries in the `ou=Accounting` branch and returns the DNs of the subentries.

```
$ bin/dump-dns -D "cn=admin,dc=example,dc=com" -w password --baseDN \
"ou=Accounting,ou=People,dc=example,dc=com" --outputFile /
usr/local/entry_dns.txt
```

2. Run the `ldapdelete` command with the file to delete the entries. The command uses the `--continueError` option, which will continue deleting through the whole list even if an error is encountered for a DN entry.

```
$ bin/ldapdelete --filename /usr/local/entry_dns.txt --continueError
```

Deleting Entries Using `ldapmodify`

You can use the LDIF `changetype` directive to delete an entry from the Directory Server using the `ldapmodify` tool. You can only delete leaf entries.

To Delete an Entry Using `ldapmodify`

- From the command line, use the `ldapmodify` tool with the `changetype:delete` directive. Enter the DN, press **Enter** to go to the next line, then enter the `changetype` directive. Press **Control-D** twice to enter the EOF sequence (UNIX) or **Control-Z** (Windows).

```
$ bin/ldapmodify --hostname server1.example.com -port 389 --bindDN
"uid=admin,dc=example,dc=com" --bindPassword password
dn:uid=user.14,ou=People,dc=example,dc=com
changetype: delete
```

Modifying Entries Using `ldapmodify`

You can use the `ldapmodify` tool to modify entries from the command line or by using an LDIF file that has the `changetype:modify` directive and value. If you have more than one change, you can separate them using the `-` (dash) symbol.

To Modify an Attribute from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changed using the `replace` directive. In this example, we change the telephone number of a specific user entry. When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Modify Multiple Attributes in an Entry from the Command Line

1. Use the `ldapsearch` tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the `ldapmodify` command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changes using the `add` and `replace` directive.

In this example, we add the `postOfficeBox` attribute, change the mobile and telephone numbers of a specific user entry. The `postOfficeBox` attribute must be present in your schema to allow the addition. The three changes are separated by a dash ("-"). When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" -w
password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postOfficeBox
postOfficeBox: 111
-
replace: mobile
mobile: +1 039 831 3737
-
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Add an Attribute from the Command Line

- Use the `ldapmodify` command from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be added using the

add directive. In this example, we add another value for the `cn` attribute, which is multi-valued. When you are done typing, you can press **CTRL-D** (UNIX EOF escape sequence) twice to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: cn
cn: Sally Tea Tree
```

An error could occur if the attribute is single-valued, if the value already exists, if the value does not meet the proper syntax, or if the value does not meet the entry's objectclass requirements. Also, make sure there are no trailing spaces after the attribute value.

To Add an Attribute Using the Language Subtype

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

```
$ bin/ldapmodify -h server.example.com -p 389 -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postalAddress; lang-ko
postalAddress; lang-ko:Byung-soon Kim$2020-14 Seoul
```

To Add an Attribute Using the Binary Subtype

The Directory Server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that could be stored in an entry. The operation must specifically match the subtype for successful operation. The version directive with a value of "1" must be used for binary subtypes. Typical binary attribute types are `userCertificate` and `jpegPhoto`.

- Use the `ldapmodify` command to add an attribute with a binary subtype. The attribute points to the file path of the certificate.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password
version: 1
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/cert
```

To Delete an Attribute

Use `ldapmodify` with the LDIF `delete` directive to delete an attribute.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password
```

```
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: employeeNumber
```

To Delete One Value from an Attribute with Multiple Values

You can use the LDIF `delete` directive to delete a specific attribute value from an attribute. For this example, assuming you have multiple values of `cn` in an entry (e.g., `cn: Sally Tree`, `cn: Sally Tea Tree`).

- Use `ldapmodify` to delete a specific attribute of a multi-valued pair, then specify the attribute pair that you want to delete. In this example, we keep `cn:Sally Tree` and delete the `cn: Sally Tea Tree`.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Sally Tea Tree
```

To Rename an Entry

Renaming an entry involves changing the relative distinguished name (RDN) of an entry. You cannot rename a RDN if it has children entries as this violates the LDAP protocol.

- Use the `ldapmodify` tool to rename an entry. The following command changes `uid=user.14` to `uid=user.2014` and uses the `changetype`, `newrdn`, and `deleteoldrdn` directives.

```
$ bin/ldapmodify
dn: uid=user.14,ou=People,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 1
```

To Move an Entry Within a Directory Server

You can use `ldapmodify` to move an entry from one base DN to another base DN. Before running the `ldapmodify` command, you must assign access control instructions (ACIs) on the parent entries. The source parent entry must have an ACI that allows export operations: `allow(export)`. The target parent entry must have an ACI that allows import operations: `allow(import)`. For more information on access control instructions, see [Working with Access Control](#).

- Use the `ldapmodify` command to move an entry from the Contractor branch to the `ou=People` branch.

```
$ bin/ldapmodify
dn: uid=user.14,ou=contractors,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 0
newsuperior: ou=People,dc=example,dc=com
```

To Move an Entry from One Machine to Another

The Directory Server provides a tool, `move-subtree`, to move a subtree or one entry on one machine to another. The subtree or entry must exist on the source server and must not be present on the target server. The source server must also support the 'real attributes only' request control. The target server must support the Ignore NO-USER-MODIFICATION request control.



Note: The `move-subtree` tool moves a subtree or multiple entries from one machine to another. The tool does not copy the entries. Once the entries are moved, they are no longer present on the source server.

- Use the `move-subtree` tool to move an entry (e.g., `uid=test.user,ou=People,dc=example,dc=com`) from the source host to the target host.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
--sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password \
--targetHost target.example.com --targetPort 389 \
--targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password \
--entryDN uid=test.user,ou=People,dc=example,dc=com
```

To Move Multiple Entries from One Machine to Another

The `move-subtree` tool provides the ability to move multiple entries listed in a DN file from one machine to another. Empty lines and lines beginning with the octothorpe character (#) will be ignored. Entry DNs may optionally be prefixed with `dn:` , but long DNs cannot be wrapped across multiple lines.

1. Open a text file, enter a list of DNs, one DN per line, and then save the file. You can also use the `ldapsearch` command with the special character "1.1" to create a file containing a list of DNs that you want to move. The following example searches for all entries that match (`department=Engineering`) and returns only the DNs that match the criteria. The results are re-directed to an output file, `test-dns.ldif`:

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
--searchScope sub "(department=Engineering)" "1.1" > test-dns.ldif
```

2. Run the `move-subtree` tool with the `--entryDNFile` option to specify the file of DNs that will be moved from one machine to another.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
--sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password \
--targetHost target.example.com --targetPort 389 \
--targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password \
--entryDNFile /path/to/file/test-dns.ldif
```

3. If an error occurs with one of the DNs in the file, the output message shows the error. The `move-subtree` tool will continue processing the remaining DNs in the file.

```
An error occurred while communicating with the target server: The entry
uid=user.2,ou=People,dc=example,dc=com cannot be added because an entry with that
name
already exists
```

```
Entry uid=user.3,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389
Entry uid=user.4,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389
```

Working with the Parallel-Update Tool

The Ping Identity Directory Server provides a `parallel-update` tool, which reads change information (add, delete, modify, and modify DN) from an LDIF file and applies the changes in parallel. This tool is a multi-threaded version of the `ldapmodify` tool that is designed to process a large number of changes as quickly as possible.

The `parallel-update` tool provides logic to prevent conflicts resulting from concurrent operations targeting the same entry or concurrent operations involving hierarchically-dependent entries (for example, modifying an entry after it has been added, or adding a child after its parent). The tool also has a retry capability that can help ensure that operations are ultimately successful even when interdependent operations are not present in the correct order in the LDIF file (for example, the change to add a parent entry is provided later in the LDIF file than a change to add a child entry).

After the tool has applied the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table

Table 17: Parallel-Update Tool Result Statistics

Processing Statistic	Description
Attempts	Number of update attempts
Successes	Number of successful update attempts
Rejects	Number of rejected updates
ToRetry	Number of updates that will be retried
AvgOps/S	Average operations per second
RctOps/S	Recent operations per second. Total number of operations from the last interval of change updates.
AvgDurMS	Average duration in milliseconds
RctDurMS	Recent duration in milliseconds. Total duration from the last interval of change updates.

To Run the Parallel-Update Tool

1. Create an LDIF file with your changes. The third change will generate a rejected entry because its `userPassword` attribute contains an encoded value, which is not allowed.

```
dn:uid=user.2,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.99,ou=People,dc=example,dc=com
changetype: moddn
newrdn: uid=user.100
deleteoldrdn: 1

dn:uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
```

```
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==

dn:uid=user.100,ou=People,dc=example,dc=com
changetype: modify
replace: st
st: TX
-
replace: employeeNumber
employeeNumber: 100
```

2. Use `parallel-update` to apply the changes in the LDIF file to a target server. In this example, we use ten concurrent threads. The optimal number of threads depends on your underlying system. The `--ldifFile` and `--rejectFile` options are also required.

```
$ bin/parallel-update --hostname 127.0.0.1 \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10
```

```
Reached the end of the LDIF file
Attempts Successes Rejects ToRetry AvgOps/S RctOps/S AvgDurMS RctDurMS
-----
          4           3           1           0           3           3           26           26
All processing complete Attempted 4 operations in 1 seconds
```

3. View the rejects file for any failed updates.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Working with the Watch-Entry Tool

The Ping Identity Directory Server provides a `watch-entry` tool, which demonstrates replication or synchronization latency by watching an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can also be directly modified with this tool as well.

To Run the Watch-Entry Tool

- Perform the following to connect to `server.example.com` as `uid=admin,dc=example,dc=com` and watch entry `uid=kate,ou=people,dc=example,dc=com` for changes:

```
$ bin/watch-entry --hostname server.example.com --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --entryDN uid=user,ou=people,dc=example,dc=com
```


Working with LDAP Transactions

The Ping Identity Directory Server provides support for *batched transactions*, which are processed all at once at commit time. Applications developed to perform batched transactions should include as few operations in the transaction as possible. The changes are not actually processed until the commit request is received. Therefore, the client cannot know whether the changes will be successful until commit time. If any of the operations fail, then the entire set of operations fails.

Batched transactions are write operations (add, delete, modify, modify DN, and password modify) that are processed as a single atomic unit when the commit request is received. If an abort request is received or an error occurs during the commit request, the changes are rolled back. The batched transaction mechanism supports the standard LDAP transaction implementation based on RFC 5805. It is not currently possible to process a transaction that requires changes to be processed across multiple servers or multiple Directory Server backends.

Directory servers may limit the set of controls that are available for use in requests that are part of a transaction. RFC 5805 section 4 indicates that the following controls may be used in conjunction with the transaction specification request control: assertion request control, manageDsaIT request control, pre-read request control, and post-read request control. The proxied authorization v1 and v2 controls cannot be included in requests that are part of a transaction, but they can be included in the start transaction request to indicate that all operations within the transaction should be processed with the specified authorization identity.

The Ping Identity Directory Server supports the following additional controls in conjunction with operations included in a transaction: account usable request control, hard delete request control, intermediate client request control, password policy request control, replication repair request control, soft delete request control, soft deleted entry access request control, subtree delete request control, and undelete request control.

To Request a Batched Transaction Using `ldapmodify`

1. Use the `ldapmodify` tool's `--useTransaction` option. It provides a mechanism for processing multiple operations as part of a single batched transaction. Create a batch text file with the changes that you want to apply as a single atomic unit:

```
dn:uid=user.3,ou=People,dc=example,dc=com
changetype: delete
dn:uid=user.1,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 383 288 1090
```

2. Use `ldapmodify` with the `--useTransaction` and `--filename` options to run the batched transaction.

```
$ bin/ldapmodify --useTransaction --filename test.ldif
```

```
#Successfully created a transaction with transaction ID 400
#Processing DELETE request for uid=user.3,ou=People,dc=example,dc=com
#DELETE operation successful for DN uid=user.3,ou=People,dc=example,dc=com
```

```
#This operation will be processed as part of transaction 400  
#Processing MODIFY request for uid=user.1,ou=People,dc=example,dc=com  
#MODIFY operation successful for DN uid=user.1,ou=People,dc=example,dc=com  
#This operation will be processed as part of transaction 400  
#Successfully committed transaction 400
```

Chapter

12 Working with Virtual Attributes

The Ping Identity Directory Server provides mechanisms to support virtual attributes in an entry. Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. While most virtual attributes are operational attributes, providing processing-related information that the server requires, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your directory server requirements.

This chapter presents the following topics:

Topics:

- [*Overview of Virtual Attributes*](#)
- [*Viewing Virtual Attribute Properties*](#)
- [*Enabling a Virtual Attribute*](#)
- [*Creating User-Defined Virtual Attributes*](#)
- [*Creating Mirror Virtual Attributes*](#)
- [*Editing a Virtual Attribute*](#)
- [*Deleting a Virtual Attribute*](#)

Overview of Virtual Attributes

The Ping Identity Directory Server allows its entries to hold virtual attributes. Virtual attributes are dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the Directory Server backend. Most virtual attributes are operational attributes, providing processing-related information that the server requires. However, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your requirements.

Viewing the List of Default Virtual Attributes

The Directory Server has a default set of virtual attributes that can be viewed using the `dsconfig` tool. Some virtual attributes are enabled by default and are useful for most applications. You can easily enable or disable each virtual attribute using the `dsconfig` tool.

The default set of virtual attributes are described in the table below. You can enable or disable these attributes using the `dsconfig` tool.

Table 18: Virtual Attributes

Virtual Attributes	Description
<code>ds-entry-checksum</code>	Generates a simple checksum of an entry's contents, which can be used with an LDAP assertion control to ensure that the entry has not been modified since it was last retrieved.
<code>ds-instance-name</code>	Generates the name of the Directory Server instance from which the associated entry was read. This virtual attribute can be useful in load-balancing environments to determine the instance from which an entry was retrieved.
<code>entryDN</code>	Generates an <code>entryDN</code> operational attribute in an entry that holds a normalized copy of the entry's current distinguished name (DN). Clients can use this attribute in search filters.
<code>hasSubordinates</code>	Creates an operational attribute that has a value of <code>TRUE</code> if the entry has subordinate entries.
<code>isMemberOf</code>	Generates an <code>isMemberOf</code> operational attribute that contains the DNs of the groups in which the user is a member.
<code>numSubordinates</code>	Generates an operational attribute that returns the number of child entries. While there is no cost if this operational attribute is enabled, there could be a performance cost if it is requested. Note that this operational attribute only returns the number of immediate children of the node.
<code>subschemaSubentry</code>	<p>A special entry that provides information in the form of operational attributes about the schema elements defined in the server. It identifies the location of the schema for that part of the tree.</p> <ul style="list-style-type: none"> > <code>IdapSyntaxes</code> - set of attribute syntaxes > <code>matchingRules</code> - set of matching rules > <code>matchingRuleUse</code> - set of matching rule uses > <code>attributeTypes</code> - set of attribute types > <code>objectClasses</code> - set of object classes

Virtual Attributes	Description
	<ul style="list-style-type: none"> > nameForms - set of name forms > dITContentRules - set of DIT content rules > dITStructureRules - set of DIT structure rules
User Defined Virtual Attribute	Generates virtual attributes with user-defined values in entries that match the criteria defined in the plug-in's configuration. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria.
Virtual Static Member	Generates a member attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the <code>Virtual Static Member</code> attribute as it is an advanced property and modifying it can lead to undesirable side effects.
Virtual Static Uniquemember	Generates a <code>uniqueMember</code> attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the <code>Virtual Static Uniquemember</code> attribute as it is an advanced property and modifying it can lead to undesirable side effects.

To View the List of Default Virtual Attributes Using dsconfig Non-Interactive Mode

- Use `dsconfig` to view the virtual attributes.

```
$ bin/dsconfig list-virtual-attributes
```

Viewing Virtual Attribute Properties

Each virtual attribute has basic properties that you can view using the `dsconfig` tool. The complete list of properties is described in the *Ping Identity Directory Server Configuration Reference*. Some basic properties are as follows:

- **Description.** A description of the virtual attribute.
- **Enabled.** Specifies whether the virtual attribute is enabled for use.
- **Base-DN.** Specifies the base DNs for the branches containing entries that are eligible to use this virtual attribute. If no values are given, the server generates virtual attributes anywhere in the server.
- **Group-DN.** Specifies the DNs of the groups whose members can use this virtual attribute. If no values are given, the group membership is not taken into account when generating the virtual attribute. If one or more group DNs are specified, then only members of those groups are allowed to have the virtual attribute.
- **Filter.** Specifies the filters that the server applies to entries to determine if they require virtual attributes. If no values are given, then any entry is eligible to have a virtual attribute value generated.

To View Virtual Attribute Properties

- Use `dsconfig` to view the properties of a virtual attribute.

```
$ bin/dsconfig get-virtual-attribute-prop --name isMemberOf
```

Enabling a Virtual Attribute

You can enable a virtual attribute using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the **Standard** object menu.

To Enable a Virtual Attribute using `dsconfig` Interactive Mode

1. Use `dsconfig` to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

```
bin/dsconfig
```

2. On the Directory Server main menu, type `o` to change the object menu, and then type the number corresponding to **Standard**.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to view and edit an existing virtual attribute.
5. From the list of existing virtual attributes on the system, select the virtual attribute to work with. For this example, type the number corresponding to the `numSubordinates` virtual attribute.
6. On the **numSubordinates Virtual Attribute Properties** menu, type the number to enable the virtual attribute. On the **Enabled Property** menu for the `numSubordinates` virtual attribute, type the number to change the value to `TRUE`.
7. On the **numSubordinates Virtual Attribute Properties** menu, type `f` to apply the changes.
8. Verify that the virtual attribute is enabled. Note that this example assumes you have configured the group entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(ou=People)" numSubordinates
```

```
dn: ou=People,dc=example,dc=com
numSubordinates: 1000
```

To Enable a Virtual Attribute Using dsconfig Non-Interactive Mode

- Use `dsconfig` to enable the `numSubordinates` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \  
--name numSubordinates --set enabled:true
```

Creating User-Defined Virtual Attributes

User-defined virtual attributes allow you to specify an explicit value to use for the virtual attribute. There are no restrictions on the length of the value for a user-defined virtual attribute. You must only ensure that the new virtual attribute conforms to your schema, otherwise you will see an error message when you configure it.

You can define your virtual attributes using the `dsconfig` tool on the Standard object menu. Only the value property is specific to the user-defined virtual attribute. All the other properties are common across all kinds of virtual attributes, which include the following:

- **enabled** -- Indicates whether the virtual attribute should be used.
- **attribute-type** -- The attribute type for the virtual attribute that will be generated.
- **base-dn, group-dn, filter** -- May be used to select which entries are eligible to contain the virtual attribute.
- **client-connection-policy** -- May be used to configure who can see the virtual values.
- **conflict-behavior** -- Used to indicate how the server should behave if there are one or more real values for the same attribute type in the same entry. The server can either return only the real value(s), only the virtual value(s), or merge both real and virtual values.
- **require-explicit-request-by-name** -- Used to indicate whether the server should only generate values for the virtual attribute if it was included in the list of requested attributes.
- **multiple-virtual-attribute-evaluation-order-index, multiple-virtual-attribute-merge-behavior** -- Used to control the behavior the server should exhibit if multiple virtual attributes may be used to contribute values to the same attribute.

To Create a User-Defined Virtual Attribute in Interactive Mode

The following example shows how to create a user-defined virtual attribute that assigns an Employee Password Policy to any entry that matches the filter `"(employeeType=employee)"`.

1. Run `dsconfig` to configure the user-defined virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

2. On the Directory Server main menu, type `o` to change the object menu, and then type the number to select Standard.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to create a new virtual attribute.
5. Next, you can use an existing virtual attribute as a template for your new attribute, or you can create a new attribute from scratch. In this example, type `n` to create a new Virtual Attribute from scratch.
6. On the **Virtual Attribute Type** menu, enter a number corresponding to the type of virtual attribute that you want to create. In this example, type the number corresponding to User Defined Virtual Attribute.
7. Next, enter a name for the new virtual attribute. In this example, enter "Employee Password Policy Assignment."
8. On the **Enabled Property** menu, enter the number to set the property to true (enable).
9. On the **Attribute-Type Property** menu, type the `attribute-type` property for the new virtual attribute. You can enter the OID number or attribute name. The `attribute-type` property must conform to your schema. For this example, type "ds-pwp-password-policy-dn".
10. Enter the value for the virtual attribute, and then press Enter or Return to continue. In this example, enter `cn=Employee Password Policy,cn=Password Policies,cn=config`, and then type Enter or Return to continue.
11. On the **User Defined Virtual Attributes** menu, enter a description for the virtual attribute. Though optional, this step is useful if you plan to create a lot of virtual attributes. Enter the option to change the value, and then type a description of the virtual attribute. In this example, enter: `Virtual attribute that assigns the Employee Password Policy to all entries that match (employeeType=employee)`.
12. On the **User Defined Virtual Attribute** menu, type the number corresponding to the filter.
13. On the **Filter Property** menu, enter the option to add one or more filter properties, type the filter, and then press **Enter** to continue. In this example, type `(employeeType=employee)`. Press the number to use the filter value entered.
14. On the **User Defined Virtual Attribute** menu, type `f` to finish creating the virtual attribute.
15. Verify that the attribute was created successfully. Add the `employeeType=employee` attribute to an entry (e.g., `uid=user.0`) using `ldapmodify`. Add the `employeeType=contractor` attribute to another entry (e.g., `uid=user.1`).
16. Use `ldapsearch` to search for the user with the `employeeType=employee` attribute (e.g., `uid=user.0`). You will notice the `ds-pwp-password-policy-dn` attribute has the assigned password policy as its value.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
```



```
ds-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Employee Password Policy,cn>Password Policies,cn=config
```

17. Run `ldapsearch` again using the filter `"(uid=user.1)"`, the `ds-pwp-password-policy-dn` attribute will not be present in the entry, because the entry has the attribute, `employeeType=contractor`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.1)" \
  ds-password-policy-dn
```

```
dn: uid=user.1,ou=People,dc=example,dc=com
```

To Create a User-Defined Virtual Attribute Using `dsconfig` in Non-Interactive Mode

- You can also use `dsconfig` in non-interactive command-line mode to create a virtual attribute. The following command sets up the Employee Password Policy Assignment virtual attribute introduced in the previous section:

```
$ bin/dsconfig create-virtual-attribute \
  --name "Employee Password Policy Assignment" \
  --type user-defined \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "filter:(employeeType=employee)" \
  --set "value:cn=Employee Password Policy,cn>Password Policies,cn=config"
```

Creating Mirror Virtual Attributes

The Ping Identity Directory Server provides a feature to mirror the value of another attribute in the same entry or mirror the value of the same or a different attribute in an entry referenced by the original entry. For example, given a DIT where users have a `manager` attributed with a value of the DN of the employee as follows:

```
dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
uid: apeters
... (more attributes) ...
```

You can set up a mirror virtual attribute, so that the returned value for the `managerName` virtual attribute can be the `cn` value of the entry referenced by the `manager` attribute as follows:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)" \
dn: uid=apeters,ou=people,dc=example,dc=com

objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
managerName: John Doe
uid: apeters
... (more attributes not shown) ...
```

To Create a Mirror Virtual Attribute in Non-Interactive Mode

You can also use `dsconfig` in non-interactive command-line mode to create a mirror virtual attribute. The following example sets up the `managerName` virtual attribute introduced in the previous section:

1. Update the schema to define the `managerName` attribute. In a text editor, create a file with the following schema definition for the attribute and save it as `98-myschema.ldif`, for example, in the `<server-root>/config/schema` folder. You can optionally add the attribute to an object class.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.4 NAME
'managerName'
EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.44{256}
X-ORIGIN 'Directory Server Example' )
```

2. Restart the Directory Server.

```
$ bin/stop-ds --restart
```

3. Use `dsconfig` to create the virtual attribute.

```
$ bin/dsconfig create-virtual-attribute \
--name "managerName" \
--type mirror \
--set "description:managerName from manager cn" \
--set enabled:true \
--set attribute-type:managerName \
--set source-attribute:cn \
--set source-entry-dn-attribute:manager
```

4. Verify the mirror virtual attribute by searching for an entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)"
```

```
dn: uid=apeters,ou=People,dc=example,dc=com
... (attributes) ...
manager: uid=jdoe,ou=People,dc=example,dc=com
managerName: John Doe
```

Editing a Virtual Attribute

You can edit virtual attributes using the `dsconfig` tool. You must ensure that the virtual attribute conforms to your plug-in schema, otherwise you will see an error message when you edit the virtual attribute. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Edit a Virtual Attribute Using dsconfig in Non-Interactive Mode

- Use `dsconfig` to change a property's value.

```
$ bin/dsconfig set-virtual-attribute-prop --name dept-number \  
--set "value:111"
```

Deleting a Virtual Attribute

You can delete virtual attributes using the `dsconfig` tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Delete a Virtual Attribute

- Use `dsconfig` to delete an existing virtual attribute.

```
$ bin/dsconfig delete-virtual-attribute --name dept-number
```


Chapter

13 Working with Groups

LDAP groups are special types of entries that represent collections of users. Groups are often used by external clients, for example, to control who has access to a particular application or features. They may also be used internally by the server to control its behavior. For example, groups can be used by the access control, criteria, or virtual attribute subsystems.

The specific ways in which clients create and interact with a particular group depends on the type of group being used. In general, there are three primary ways in which clients attempt to use groups:

- > To determine whether a specified user is a member of a particular group.
- > To determine the set of groups in which a specified user is a member.
- > To determine the set of all users that are members of a particular group.

This chapter provides an overview of Directory Server groups concepts and provides procedures on setting up and querying groups in the Directory Server.

Topics:

- [Overview of Groups](#)
- [About the `isMemberOf` and `isDirectMemberOf` Virtual Attribute](#)
- [Using Static Groups](#)
- [Using Dynamic Groups](#)
- [Using Virtual Static Groups](#)
- [Creating Nested Groups](#)
- [Maintaining Referential Integrity with Static Groups](#)
- [Monitoring the Group Membership Cache](#)
- [Using the Entry Cache to Improve the Performance of Large Static Groups](#)
- [Tuning the Index Entry Limit for Large Groups](#)
- [Summary of Commands to Search for Group Membership](#)
- [Migrating Sun/Oracle Groups](#)

Overview of Groups

The Directory Server provides the following types of groups:

- **Static Groups.** A static group is an entry that contains an explicit list of member or `uniquemember` attributes, depending on its particular structural object class. Static groups are ideal for relatively small, infrequently changing elements. Once the membership list grows, static groups become more difficult to manage as any change in a member base DN must also be changed in the group. Static groups use one of three structural object classes: `groupOfNames`, `groupOfUniqueNames`, and `groupOfEntries`.

The Directory Server also supports nested groups, in which a parent group entry contains child attributes whose DNs reference another group. Nested groups are a flexible means to organize entries that provide inherited group membership and privileges. To maintain good performance throughout, a group cache is enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups.

- **Dynamic Groups.** A dynamic group has its membership list determined by search criteria using a LDAP URL. Dynamic groups solve the scalability issues encountered for static groups as searches are efficient, constant-time operations. However, if searches range over a very large set of data, performance could be affected.
- **Virtual Static Groups.** A virtual static group is a combination of both static and dynamic groups, in which each member in a group is a virtual attribute that is dynamically generated when invoked. Virtual static groups solve the scalability issues for clients that can only support static groups and are best used when the application targets a search operation for a specific member. Virtual static groups are not good for applications that need to retrieve the entire membership list as the process for constructing the entire membership list can be expensive.

About the `isMemberOf` and `isDirectMemberOf` Virtual Attribute

The existence of both static, nested, dynamic, and virtual static groups can make it unnecessarily complex to work with groups in the server, particularly because the ways you interact with them are so different. And the fact that static groups can use three different structural object classes (not counting the auxiliary class for virtual static groups) does not make things any easier.

To make group operations simpler, the Ping Identity Directory Server provides the ability to generate either an `isMemberOf` and `isDirectMemberOf` virtual attributes in user entries. These attributes dramatically simplify the process for making group-related determinations in a manner that is consistent across all types of groups.

The value of the `isMemberOf` virtual attribute is a list of DNs of all groups (including static, nested, dynamic, and virtual static groups) in which the associated user is a member. The value of the `isDirectMemberOf` virtual attribute is a subset of the values of `isMemberOf`, which represents the groups for which the entry is an explicit or direct member. While the `isMemberOf`

virtual attribute is enabled by default, you must enable the `isDirectMemberOf` virtual attribute if you plan to use it. Run the following `dsconfig` command to enable the virtual attribute:

```
dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set enabled:true
```

Because the `isMemberOf` and `isDirectMemberOf` are operational attributes, only users who specifically have been granted the privilege can see it. The default set of access control rules do not allow any level of access to user data. The only access that is granted is what is included in user-defined access control rules, which is generally given to a `uid=admin` administrator account. It is always a best practice to restrict access to operational and non-operational attributes to the minimal set of users that need to see them. The root bind DN, `cn=Directory Manager`, has the privilege to view operational attributes by default.

To determine whether a user is a member of a specified group using the `isMemberOf` virtual attribute, simply perform a base-level search against the user's entry with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group. The `isMemberOf` attribute is best implemented when you want to only find groups that users are an actual member of (i.e., not including nested group membership). The following table illustrates this simple base-level search:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	1.1

If this search returns an entry, then the user is a member of the specified group. If no entry is returned, then the user is not a member of the given group.

To determine the set of all groups in which a user is a member, simply retrieve the user's entry with a base-level search and include the `isMemberOf` attribute:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(objectclass=*)
Requested Attributes	isMemberOf

To determine the set of all members for a specified group, issue a subtree search with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group and requesting the attributes you wish to have for member entries:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	sub
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	cn, mail

The `isDirectMemberOf` virtual attribute can be used in the examples above in place of `isMemberOf` if you only need to find groups that users are an actual member of. You must use `isMemberOf` for nested group membership.

Note that if this filter targets a dynamic group using an unindexed search, then this may be an expensive operation. However, it will not be any more expensive than retrieving the target group and then issuing a search based on information contained in the member URL.

For static groups, this approach has the added benefit of using a single search to retrieve information from all user entries, whereas it would otherwise be required to retrieve the static group and then perform a separate search for each member's entry.

Using Static Groups

A static group contains an explicit membership list where each member is represented as a DN-valued attribute. There are three types of static groups supported for use in the Directory Server:

- **groupOfNames.** A static group that is defined with the `groupOfNames` structural object class and uses the `member` attribute to hold the DNs of its members. RFC 4519 requires that the `member` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `member` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfUniqueNames.** A static group that is defined with the `groupOfUniqueNames` structural object class and uses the `uniquemember` attribute to hold the DNs of its members. RFC 4519 requires that the `uniquemember` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `uniquemember` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfUniqueNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Test Group
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
uniquemember: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfEntries.** A static group that is defined with the `groupOfEntries` object class and uses the `member` attribute to hold the DNs of its members. This group specifies that the `member` attribute is optional to ensure that the last member can be removed from the group. Although the draft proposal (`draft-findlay-ldap-groupofentries-00.txt`) has expired, the Directory Server supports this implementation. The following entry depicts a group defined with the `groupOfEntries` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfEntries
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```


Creating Static Groups

You can configure a static group by adding it using an LDIF file. Static groups contain a membership list of explicit DNs specified by the `uniquemember` attribute.

To Create a Static Group

1. Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniqueNames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `static-group.ldif`. The following example LDIF file creates two groups, `cn=Development` and `cn=QA`.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Development,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Development
ou: groups
uniquemember: uid=user.14,ou=People,dc=example,dc=com
uniquemember: uid=user.91,ou=People,dc=example,dc=com
uniquemember: uid=user.180,ou=People,dc=example,dc=com

dn: cn=QA,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: QA
ou: groups
uniquemember: uid=user.0,ou=People,dc=example,dc=com
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename static-group.ldif
```

3. Verify the configuration by using the virtual attribute `isDirectMemberOf` that checks membership for a non-nested group. By default, the virtual attribute is disabled by default, but you can enable it using `dsconfig`.

```
$ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set enabled:true
```

4. Use `ldapsearch` to specifically search the `isDirectMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf
```

```
dn: uid=user.14,ou=People,dc=example,dc=com
isDirectMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

5. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an `aci` attribute in an LDIF file, and save the file as `dev-group-aci.ldif`.

Add the file using the `ldapmodify` tool. You can create a similar ACI for the QA group, which is not shown in this example.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com" )
      (targetattr != "cn || sn || uid")
      (targetfilter = "(ou=Development)")
      (version 3.0; acl "Dev Group Permissions";
        allow (write) (groupdn = "ldap:///cn=Development,ou=groups,dc=example,dc=com");)
```

6. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename dev-group-aci.ldif
```

To Add a New Member to a Static Group

- To add a new member to the group, add a new value for the `uniquemember` attribute that specifies the DN of the user to be added. The following example adds a new `uniquemember`, `user.4`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
add: uniquemember
uniquemember: uid=user.4,ou=People,dc=example,dc=com
```

To Remove a Member from a Static Group

- To remove a member from a static group, remove that user's DN from the `uniquemember` attribute. The following example removes the DN of `user.1`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
delete: uniquemember
uniquemember: uid=user.1,ou=People,dc=example,dc=com
```

Searching Static Groups

The following sections describe how to compose searches to determine if a user is a member of a static group, to determine all the static groups in which a user is a member, and to determine all the members of a static group.

To Determine if a User is a Static Group Member

To determine whether a user is a member of a specified group, perform a base-level search to retrieve the group entry with an equality filter looking for the membership attribute of a value equal to the DN of the specified user.

For best performance, you will want to include a specific attribute list (just "cn", or "1.1" request that no attributes be returned) so that the entire member list is not returned. For example, to determine whether the user "`uid=john.doe,ou=People,dc=example,dc=com`" is a member of the `groupOfNames` static group "`cn=Test Group,ou=Groups,dc=example,dc=com`", issue a search with the following criteria:

Table 19: Search Criteria for a Single User's Membership in a Static Group

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(member=uid=john.doe,ou=People,dc=example,dc=com)
Requested Attributes	1.1

If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group. If you do not know the membership attribute for the specified group (it could be either a `member` or `uniqueMember` attribute), then you may want to revise the filter so that it allows either one as follows:

```
( |(member=uid=john.doe,ou=People,dc=example,dc=com)
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com) )
```

- Run a base-level search to retrieve the group entry with an equality filter looking for the membership attribute.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com"
--searchScope base "(member=uid=john.doe,ou=People,dc=example,dc=com)" "1.1"
```

To Determine the Static Groups to Which a User Belongs

To determine the set of all static groups in which a user is specified as a member, perform a subtree search based at the top of the DIT. The search filter must be configured to match any type of static group in which the specified user is a member.

For example, the following criteria may be used to determine the set of all static groups in which the user, `uid=john.doe,ou=People,dc=example,dc=com`, is a member:

Table 20: Search Criteria for Determining All the Static Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(((&(objectClass=groupOfNames) (member=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfUniqueNames)(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfEntries) (member=uid=john.doe,ou=People,dc=example,dc=com))))
Requested Attributes	1.1

Every entry returned from the search represents a static group in which the specified user is a member.

- Run a sub-level search to retrieve the static groups to which a user belongs.

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" --searchScope sub \
" ( | (&(objectClass=groupOfNames)
(member=uid=john.doe,ou=People,dc=example,dc=com)) \
(&(objectClass=groupOfUniqueNames) \
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) \
(&(objectClass=groupOfEntries) \
(member=uid=john.doe,ou=People,dc=example,dc=com)) ) " "1.1"
```



Note: A base level search of the user's entry for `isMemberOf` or `isDirectMemberOf` virtual attributes will give the same results. You can also use the virtual attributes with virtual static groups.

To Determine the Members of a Static Group

To determine all of the members for a static group, simply retrieve the group entry including the membership attribute. The returned entry will include the DNs of all users that are members of that group. For example, the following criteria may be used to retrieve the list of all members for the group `cn=Test Group,ou=Groups,dc=example,dc=com`:

Table 21: Search Criteria for All of the Static Group's Members

Base DN	<code>cn=Test Group,ou=Groups,dc=example,dc=com</code>
Scope	<code>base</code>
Filter	<code>(objectClass=*)</code>
Requested Attributes	<code>member uniqueMember</code>

If you want to retrieve additional information about the members, such as attributes from member entries, you must issue a separate search for each member to retrieve the user entry and the desired attributes.

- Run a base-level search to retrieve all of the members in a static group.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com" \
  --searchScope base "(objectclass=*)" uniqueMember
```



Note: If you want to retrieve attributes from member entries, it is more efficient to search all users whose `isMemberOf` attribute contains the group DN, returning the attributes desired.

Using Dynamic Groups

Dynamic groups contain a set of criteria used to identify members rather than maintaining an explicit list of group members. If a new user entry is created or if an existing entry is modified so that it matches the membership criteria, then the user will be considered a member of the dynamic group. Similarly, if a member's entry is deleted or if it is modified so that it no longer matches the group criteria, then the user will no longer be considered a member of the dynamic group.

In the Directory Server, dynamic groups include the `groupOfURLs` structural object class and use the `memberurl` attribute to provide an LDAP URL that defines the membership criteria. The base, scope, and filter of the LDAP URL will be used in the process of making the determination, and any other elements present in the URL will be ignored. For example, the

following entry defines a dynamic group in which all users below `dc=example,dc=com` with an `employeeType` value of `contractor` will be considered members of the group:

```
dn: cn=Sales Group,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Sales Group
memberURL: ldap:///dc=example,dc=com??sub?(employeeType=contractor)
```

Assuming that less than 80,000 entries have the `employeeType` of `contractor`, you need to create the following index definition to evaluate the dynamic group:

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
--index-name employeeType --set index-entry-limit:80000 \
--set index-type:equality
```

Creating Dynamic Groups

You can configure a dynamic group in the same manner as static groups using an LDIF file. Dynamic groups contain a membership list of attributes determined by search filter using an LDAP URL. You must use the `groupOfURLs` object class and the `memberURL` attribute.

To Create a Dynamic Group

1. Assume that `uid=user.15` is not part of any group. Use `ldapsearch` to verify that `uid=user.15` is not part of any group. In a later step, we will add the user to the dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.15)" ou
dn: uid=user.15,ou=People,dc=example,dc=com
```

2. Assume for this example that `uid=user.0` has an `ou=Engineering` attribute indicating that he or she is a member of the Engineering department.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)" ou
isMemberOf
dn: uid=user.0,ou=People,dc=example,dc=com
ou: Engineering
```

3. Open a text editor, and then create a dynamic group entry in LDIF. The LDIF defines the dynamic group to include all users who have the `ou=Engineering` attribute. When done, save the file as `add-dynamic-group.ldif`.

```
dn: cn=eng-staff,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfURLs
ou: groups
cn: eng-staff
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(ou=Engineering)
```

4. Use `ldapmodify` to add the group entry to the server.

```
$ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
```

5. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.0` is a member of the `cn=Engineering` group or any other group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

6. If your data is relatively small (under 1 million entries), you can search for all users in the group that meet the search criteria (`ou=Engineering`). For very large databases, it is not practical to run a database-wide search for all users as there can be a performance hit on the Directory Server. The following command returns the DNs of entries that are part of the `cn=eng-staff` dynamic group and sorts them in ascending order by the `sn` attribute.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
"(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
```

7. Add `uid=user.15` to the `eng-staff` group by adding an `ou=Engineering` attribute to the entry. This step highlights an advantage of dynamic groups: you can make a change in an entry without explicitly adding the DN to the group as you would with static groups. The entry will be automatically added to the `eng-staff` dynamic group.

```
$ bin/ldapmodify
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: modify
add: ou
ou: Engineering
```

8. Use `ldapsearch` to check if the user is part of the `cn=eng-staff` dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.15)"
isMemberOf
dn: uid=user.15,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

Searching Dynamic Groups

The following sections describe how to compose searches to determine if a user is a member of a dynamic group, to determine all the dynamic groups in which a user is a member, and to determine all the members of a dynamic group.

To Determine if a User is a Dynamic Group Member

To determine whether a user is a member of a specific dynamic group, you must verify that the user's entry is both within the scope of the member URL and that it matches the filter contained in that URL. You can verify that a user's entry is within the scope of the URL using simple client-side only processing. Evaluating the filter against the entry on the client side can be more complicated. While possible, particularly in clients that are able to perform schema-aware evaluation, a simple alternative is to perform a base-level search to retrieve the user's entry with the filter contained in the member URL.

For example, to determine whether the user `uid=john.doe,ou=People,dc=example,dc=com` is a member of the dynamic group with the above member URL, issue a search with the following criteria:

Table 22: Search Criteria for a Single User's Membership in a Dynamic Group

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(ou=Engineering)
Requested Attributes	1.1

Note that the search requires the user DN to be under the search base defined in the `memberurl` attribute for the user to be a member. If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group.

To Determine the Dynamic Groups to Which a User Belongs

To determine the set of all dynamic groups in which a user is a member, first perform a search to find all dynamic group entries defined in the server. You can do this using a subtree search with a filter of `"(objectClass=groupOfURLs)"`.

You should retrieve the `memberURL` attribute so that you can use the logic described in the previous section to determine whether the specified user is a member of each of those groups. For example, to find the set of all dynamic groups defined in the `dc=example,dc=com` tree, issue a search with the following criteria:

Table 23: Search Criteria for Determining All of the Dynamic Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(objectClass=groupOfURLs)
Requested Attributes	memberURL

Each entry returned will be a dynamic group definition. You can use the base, scope, and filter of its `memberURL` attribute to determine whether the user is a member of that dynamic group.

To Determine the Members of a Dynamic Group

To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL. The set of requested attributes should reflect the attributes desired from the member user entries, or "1.1" if no attributes are needed.

For example, to retrieve the `cn` and `mail` attributes from the group described above, use the following search:

Table 24: Search Criteria for Determining the Members of a Dynamic Group

Base DN	dc=example,dc=com
Scope	sub
Filter	(employeeType=contractor)
Requested Attributes	cn, mail



Caution: Note that this search may be expensive if the associated filter is not indexed or if the group contains a large number of members.

Using Dynamic Groups for Internal Operations

You can use dynamic groups for internal operations, such as ACI or component evaluation. The Directory Server performs the `memberurl` parsing and internal LDAP search; however, the internal search operation may not be performed with access control rules applied to it.

For example, the following dynamic group represents an organization's employees within the same department:

```
dn: cn=department 202,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: department 202
owner: uid=user.1,ou=people,dc=example,dc=com
owner: uid=user.2,ou=people,dc=example,dc=com
memberURL: ldap:///ou=People,dc=example,dc=com??sub?
(&(employeeType=employee)(departmentNumber=202))
description: Group of employees in department 202
```

The above group could be referenced from within the ACI at the `dc=example,dc=com` entry. For example:

```
dn:dc=example,dc=com
aci: (targetattr="employeeType")
(version 3.0; acl "Grant write access to employeeType" ;
allow (all) groupdn="ldap:///cn=department 202,ou=groups,dc=example,dc=com";)
```

Any user matching the filter can bind to the server with their entry and modify the `employeeType` attribute within any entry under `dc=example,dc=com`.

Using Virtual Static Groups

Static groups can be easier to interact with than dynamic groups, but large static groups can be expensive to manage and require a large amount of memory to hold in the internal group cache. The Directory Server provides a third type of group that makes it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with it as a static group. A *virtual static group* is a type of group that references another group and provides access to the members of that group as if it was a static group.

To create a virtual static group, create an entry that has a structural object class of either `groupOfNames` or `groupOfUniqueNames` and an auxiliary class of `ds-virtual-static-group`. It should also include a `ds-target-group-dn` attribute, whose value is the group from which the virtual static group should obtain its members. For example, the following will create a virtual static group that exposes the members of the `cn=Sales Group,ou=Groups,dc=example,dc=com` dynamic group as if it were a static group:

```
dn: cn=Virtual Static Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static Sales Group
```



```
ds-target-group-dn: cn=Sales Group,ou=Groups,dc=example,dc=com
```

Note that you must also enable a virtual attribute that allows the `member` attribute to be generated based on membership for the target group. A configuration object for this virtual attribute does exist in the server configuration, but is disabled by default. To enable it, issue the following change:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set enabled:true
```

If you want to use virtual static groups with the `groupOfUniqueNames` object class, then you will also need to enable the `Virtual Static uniqueMember` virtual attribute in the same way.

Creating Virtual Static Groups

If your application only supports static groups but has scalability issues, then using a virtual static group could be a possible solution. A virtual static group uses a virtual attribute that is dynamically generated when called after which the operations that determine group membership are passed to another group, such as a dynamic group. You must use the `ds-virtual-static-group` object class and the `ds-target-group-dn` virtual attribute.

Virtual static groups are best used when determining if a single user is a member of a group. It is not a good solution if an application accesses the full list of group members due to the performance expense at constructing the list. If you have a small database and an application that requires that the full membership list be returned, you must also enable the `allow-retrieving-membership` property for the `Virtual Static uniqueMember` virtual attribute using the `dsconfig` tool.

To Create a Virtual Static Group

1. Open a text editor, and then create a group entry in LDIF. The entry contains the `groupOfUniqueNames` object class, but in place of the `uniqueMember` attribute is the `ds-target-group-dn` virtual attribute, which is part of the `ds-virtual-static-group` auxiliary object class. When done, save the file as `add-virtual-static-group.ldif`.

```
dn: cn=virtualstatic,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
objectclass: ds-virtual-static-group
ou: groups
cn: virtual static
ds-target-group-dn: cn=eng-staff,ou=groups,dc=example,dc=com
```

2. Use `ldapmodify` to add the virtual static group entry to the server.

```
$ bin/ldapmodify -h server1.example.com -p 389 -D "uid=admin,dc=example,dc=com" \
  -w password -a -f add-virtual-static-group.ldif
```

3. Use `dsconfig` to enable the `Virtual Static uniqueMember` attribute, which is disabled by default.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
  --set enabled:true
```

4. In the previous section, we set up `uid=user.0` to be part of the `cn=eng-staff` dynamic group. Use `ldapsearch` with the `isMemberOf` virtual attribute to determine if `uid=user.0` is part of the virtual static group.

```
$ bin/ldapsearch -h server1.example.com -p 389 -D "cn=Directory Manager" \
-w secret -b dc=example,dc=com "(uid=user.0)" isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=virtualstatic,ou=groups,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

5. Use `ldapsearch` to determine if `uid=user.0` is a member of the virtual static group. You should see the returned `cn=virtualstatic` entry if successful.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
-b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
"(&(objectclass=groupOfUniqueNames) \
(uniqueMember=uid=user.0,ou=People,dc=example,dc=com))"
```

6. Next, try searching for a user that is not part of the `cn=eng-staff` dynamic group (e.g., `uid=user.20`), nothing will be returned.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
-b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
"(&(objectclass=groupOfUniqueNames) \
(uniqueMember=uid=user.20,ou=People,dc=example,dc=com))"
```

Searching Virtual Static Groups

Because virtual static groups behave like static groups, the process for determining whether a user is a member of a virtual static group is identical to that of a member in a static group. Similarly, the process for determining all virtual static groups in which a user is a member is basically the same as the process as that of real static groups in which a user is a member. In fact, the query provided in the static groups discussion returns virtual static groups in addition to real static groups, because the structural object class of a virtual static group is the same as the structural object class for a static group.

You can also retrieve a list of all members of a virtual static group in the same way as a real static group: simply retrieve the `member` or `uniqueMember` attribute of the desired group. However, because virtual static groups are backed by dynamic groups and the process for retrieving member information for dynamic groups can be expensive, virtual static groups do not allow retrieving the full set of members by default. The virtual attribute used to expose membership can be updated to allow this with a configuration change such as the following:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
--set allow-retrieving-membership:true
```

Because this can be an expensive operation, we recommend that the option to allow retrieving virtual static group membership be left disabled unless it is required.

Creating Nested Groups

The Ping Identity Directory Server supports nested groups, where the DN of an entry that defines a group is included as a member in the parent entry. For example, the following example shows a nested static group (e.g., `cn=Engineering Group`) that has `uniquemember` attributes consisting of other groups, such as `cn=Developers Group` and the `cn=QA Group` respectively.

```
dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=Groups,dc=example,dc=com
uniquemember: cn=QA,ou=Groups,dc=example,dc=com
```

Nested group support is enabled by default on the Directory Server. To support nested groups without the performance hit, the Directory Server uses a group cache, which is also enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups. The Directory Server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

In practice, nested groups are not commonly used for several reasons. LDAP specifications do not directly address the concept of nested groups, and some servers do not provide any level of support for them. Supporting nested groups in LDAP clients is not trivial, and many directory server-enabled applications that can interact with groups do not provide any support for nesting. If nesting support is not needed in your environment, or if nesting support is only required for clients but is not needed for server-side evaluation (such as for groups used in access control rules, criteria, virtual attributes, or other ways that the server may need to make a membership determination), then this support should be disabled.

To Create Nested Static Groups

1. The following example shows how to set up a nested static group, which is a static group that contains `uniquemember` attributes whose values contain other groups (static, virtual static, or dynamic). Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniqueNames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `nested-group.ldif`. Assume that the static groups, `cn=Developers Group` and `cn=QA Group`, have been configured.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Engineering Group,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=groups,dc=example,dc=com
uniquemember: cn=QA,ou=groups,dc=example,dc=com
```

2. Use `ldapmodify` to add the group entry.

```
$ bin/ldapmodify --defaultAdd --filename nested-static-group.ldif
```

3. Verify the configuration by using the `isMemberOf` virtual attribute that checks the group membership for an entry. By default, the virtual attribute is enabled. Use `ldapsearch` to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isMemberOf
dn: uid=user.14,ou=People,dc=example,dc=com
isMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

4. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an ACI in LDIF, and save the file as `eng-group-aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com")
      (targetattr != "cn || sn || uid")
      (targetfilter = "(ou=Engineering Group)")
      (version 3.0; acl "Engineering Group Permissions";
       allow (write) (groupdn = "ldap:///cn=Engineering
       Group,ou=groups,dc=example,dc=com");)
```

5. Add the file using the `ldapmodify` tool.

```
$ bin/ldapmodify --filename eng-group-aci.ldif
```

Note: When nesting dynamic groups, you cannot include other groups as members of a dynamic group. You can only support "nesting" by including the members of another group with a filter in the member URL. For example, if you have two groups `cn=dynamic1` and `cn=dynamic2`, you can nest one group in another by specifying it in the member URL as follows:



```
cn=dynamic1,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
memberURL: ldap:///dc=example,dc=com??sub?
(isMemberOf=cn=dynamic2,ou=groups,dc=example,dc=com)
```

The members included from the other group using this method are not considered "nested" members and will be returned even when using `isDirectMemberOf` when retrieving the members.

Maintaining Referential Integrity with Static Groups

The Directory Server can automatically update references to an entry whenever that entry is removed or renamed in a process called *referential integrity*. For example, if a user entry is deleted, then referential integrity plugin will remove that user from any static groups in which the user was a member (this is not necessary for dynamic groups, since no explicit membership is maintained). Similarly, if a modify DN operation is performed to move or rename a user

entry, then referential integrity updates static groups in which that user is a member with the new user DN.

Referential integrity support is disabled by default, but may be enabled using the `dsconfig` tool as follows:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \  
--set enabled:true
```

Other configuration attributes of note for this plugin include:

- **attribute-type.** This attribute specifies the names or OIDs of the attribute types for which referential integrity will be maintained. By default, referential integrity is maintained for the `member` and `uniqueMember` attributes. Any attribute types specified must have a syntax of either distinguished name (OID "1.3.6.1.4.1.1466.115.121.1.12") or name and optional UID (OID "1.3.6.1.4.1.1466.115.121.1.34"). The specified attribute types must also be indexed for equality in all backends for which referential integrity is to be maintained.
- **base-dn.** This attribute specifies the subtrees for which referential integrity will be maintained. If one or more values are provided, then referential integrity processing will only be performed for entries which exist within those portions of the DIT. If no values are provided (which is the default behavior), then entries within all public naming contexts will be included.
- **log-file.** This attribute specifies the path to a log file that may be used to hold information about the DNs of deleted or renamed entries. If the plugin is configured with a nonzero update interval, this log file helps ensure that appropriate referential integrity processing occurs even if the server is restarted.
- **update-interval.** This attribute specifies the maximum length of time that a background thread may sleep between checks of the referential integrity log file to determine whether any referential integrity processing is required. By default, this attribute has a value of "0 seconds", which indicates that all referential integrity processing is to be performed synchronously before a response is returned to the client. A duration greater than 0 seconds indicates that referential integrity processing will be performed in the background and will not delay the response to the client.

In the default configuration, where referential integrity processing is performed synchronously, the throughput and response time of delete and modify DN operations may be adversely impacted because the necessary cleanup work must be completed before the response to the original operation can be returned. Changing the configuration to use a non-zero update interval alleviates this performance impact because referential integrity processing uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

However, performing referential integrity processing in a background thread may introduce a race condition that may adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. If referential integrity processing has not yet been completed for the delete, then newly-established group memberships may be removed along with those that already existed for the previous user. Similarly, if the newly-created user is to be a member of one or more of the same groups as the previous user, then attempts by the client to re-establish those memberships may fail if referential integrity processing has not yet removed the previous membership. For this reason, we recommend that the default synchronous behavior be maintained unless the performance impact associated with it is unacceptable and

clients are not expected to operate in a manner that may be adversely impacted by delayed referential integrity processing.



Note: The internal operations of the referential integrity plug-in are not replicated. So, in a replicated topology, you must enable the referential integrity plug-in consistently on all servers in the topology to ensure that changes made by the referential integrity plug-in are passed along to a replication server.

For more information about administering the referential integrity plug-in, see Chapter 6, “Configuring the Directory Server” in the *PingData Directory Server Administration Guide*.

Monitoring the Group Membership Cache

The Directory Server logs information at startup about the memory consumed by the group membership cache. This hard-coded cache contains information about all of the group memberships for internal processing, such as ACIs. The group membership cache is enabled by default.

The information about this cache is logged to the standard output log (`server.out`) and the standard error log. When using groups, you can use the log information to tune the server for best performance. For example, at startup the server logs a message like the following to the `server.out` log:

```
[16/Aug/2011:17:14:39.462 -0500] category=JEB severity=NOTICE msgID=1887895587
msg="The database cache now holds 3419MB of data and is 32 percent full"
```

The error log will contain something like the following:

```
[16/Aug/2011:18:40:39.555 -0500] category=EXTENSIONS severity=NOTICE msgID=1880555575
msg="'Group cache (174789 static group(s) with 7480151 total memberships and 1000002
unique members, 0 virtual static group(s), 1 dynamic group(s))' currently consumes
149433592 bytes and can grow to a maximum of 149433592 bytes"
```

Using the Entry Cache to Improve the Performance of Large Static Groups

The Ping Identity Directory Server provides an entry cache implementation, which allows for fine-grained control over the kinds of entries that may be held in the cache. You can define filters to specify the entries included in or excluded from the cache, and you can restrict the cache so that it holds only entries with at least a specified number of values for a given set of attributes.

Under most circumstances, we recommend that the Directory Server be used *without* an entry cache. The Directory Server is designed to efficiently retrieve and decode entries from the database in most cases. The database cache is much more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

However, if the Directory Server contains very large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, then you may want to enable an entry cache that holds only large static groups.

In servers containing large static groups, you can define an entry cache to hold only those large static groups. This entry cache should have an include filter that matches only group entries (for example, "`(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfEntries))`"). The filter contains a minimum value count so that only groups with a large number of members (such as those with at least 100 `member` or `uniqueMember` values) will be included. The Directory Server provides an entry cache implementation with these settings although it is disabled by default.

To Enable the Entry Cache

- Run `dsconfig` to enable the entry cache.

```
$ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache" \
--set enabled:true
```

To Create Your Own Entry Cache for Large Groups

- You can create your own entry cache for large groups using the `dsconfig create-entry-cache` subcommand.

```
# bin/dsconfig create-entry-cache --type fifo \
--set enabled:true \
--set cache-level:10 \
--set max-entries:175000 \
--set "include-filter:(objectClass=groupOfUniqueNames)" \
--set min-cache-entry-value-count:10000 \
--set min-cache-entry-attribute:uniquemember
```

Monitoring the Entry Cache

You can monitor the memory consumed by your entry cache using the `entry-cache-info` property in the periodic stats logger. You can retrieve the monitor entry over LDAP by issuing a search on `baseDN="cn=monitor"` using `filter="(objectClass=ds-fifo-entry-cache-monitor-entry)"`. For example, the entry might appear as follows:

```
dn: cn=Static Group Entry Cache Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-fifo-entry-cache-monitor-entry
objectClass: extensibleObject
cn: Static Group Entry Cache Monitor
cacheName: Static Group Entry Cache
entryCacheHits: 6416407
entryCacheTries: 43069073
entryCacheHitRatio: 14
maxEntryCacheSize: 12723879900
currentEntryCacheCount: 1
maxEntryCacheCount: 175000
entriesAddedOrUpdated: 1
evictionsDueToMaxMemory: 0
evictionsDueToMaxEntries: 0
entriesNotAddedAlreadyPresent: 0
```

```
entriesNotAddedDueToMaxMemory: 0
entriesNotAddedDueToFilter: 36652665
entriesNotAddedDueToEntrySmallness: 0
lowMemoryOccurrences: 0
percentFullMaxEntries: 0
jvmMemoryMaxPercentThreshold: 75
jvmMemoryCurrentPercentFull: 24
jvmMemoryBelowMaxMemoryPercent: 51
isFull: false
capacityDetails: NOT FULL: The JVM is using 24% of its available memory. Entries can be
added to the cache until the overall JVM memory usage reaches the configured limit of
75%. Cache has 174999 remaining entries before reaching the configured limit of 175000.
```

By default, the entry cache memory is set to 75%, with a maximum of 90%.

Tuning the Index Entry Limit for Large Groups

The Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the DIT. You can specify an index entry limit property, which defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (which is 4000 by default), then you must rebuild the indexes using the `rebuild-index` tool as follows:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
```

In the majority of Directory Server environments, the default index entry limit value of 4000 entries should be sufficient. However, group-related processing, it may be necessary to increase the index entry limit. For directories containing more than 4000 groups with the same structural object class (i.e., more than 4000 entries, 4000 `groupOfUniqueNames` entries, 4000 `groupOfEntries` entries, or 4000 `groupOfURLs` entries), then you may want to increase the index entry limit for the `objectClass` attribute so that it has a value larger than the maximum number of group entries of each type. Set `index-entry-limit` property using a command line like the following:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
--index-name objectClass --set index-entry-limit:175000
```

As an alternative, a separate backend may be created to hold these group entries, so that an unindexed search in that backend yields primarily group entries. If you make no changes, then the internal search performed at startup to identify all groups and any user searches looking for groups of a given type may be very expensive.

For directories in which any single user may be a member of more than 4000 static groups of the same type, you may need to increase the index entry limit for the `member` and/or `uniqueMember` attribute to a value larger than the maximum number of groups in which any user is a member. If you do not increase the limit, then searches to retrieve the set of all static groups in which the user is a member may be unindexed and therefore very expensive.

Summary of Commands to Search for Group Membership

The following summary of commands show the fastest way to retrieve direct or indirect member DNs for groups.

- To retrieve direct member (non-nested) DNs of group "cn=group.1,ou=groups,dc=example,dc=com".


```
$ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com" "(objectClass=*)"
uniqueMember member
```
- To retrieve direct member entries (non-nested) under "dc=example,dc=com" of group "cn=group.1,ou=groups,dc=example,dc=com". This is useful when attributes from member entries are used in the filter or being returned.


```
$ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
"(isDirectMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
```
- To retrieve group DNs in which user "uid=user.2,ou=people,dc=example,dc=com" is a direct member (non-nested, static groups).


```
$ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com" "(objectClass=*)"
isDirectMemberOf
```
- To retrieve all member entries under ou=people,dc=example,dc=com of group "cn=group.1,ou=groups,dc=example,dc=com".


```
$ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
"(isMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
```
- To retrieve the group DNs in which user "uid=user.2,ou=people,dc=example,dc=com" is a member.


```
$ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com" "(objectClass=*)"
isMemberOf
```

Migrating Sun/Oracle Groups

You can migrate Sun/Oracle static and dynamic groups to Ping Identity Directory Server groups. The following sections outline the procedures for migrating static groups to both PingData static groups and virtual static groups as well as how to migrate dynamic groups. For information about the differences in access control evaluation between Sun/Oracle and the Ping Identity Directory Server, see [Migrating ACIs from Sun/Oracle to Ping Identity Directory Server](#).

Migrating Static Groups

The Ping Identity Directory Server supports static LDAP groups with structural object classes of `groupOfNames`, `groupOfUniqueNames`, or `groupOfEntries`. In general, static groups may be imported without modification.

A FIFO entry cache can be enabled to cache group-to-user mappings, which improves performance when accessing very large entries, though at the expense of greater memory consumption. The Ping Identity Directory Server provides an out-of-the-box FIFO entry cache object for this purpose. This object must be explicitly enabled using `dsconfig` as described in [Using the Entry Cache to Improve the Performance of Large Static Groups](#).

To Migrate Static Groups

1. Run the `migrate-ldap-schema` tool to enumerate any schema differences between the DSEE deployment and the PingData deployment.
2. Run the `migrate-sun-ds-config` tool to enumerate any configuration differences between the DSEE deployment and the PingData deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the `import-ldif` tool.
5. From the Ping Identity Directory Server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder using a text editor.
6. Find the **FIFO Entry Cache** section and, after reading the accompanying comments, enable the corresponding `dsconfig` command by removing the comment character ("`#`").

```
$ bin/dsconfig set-entry-cache-prop \  
--cache-name "Static Group Entry Cache" --set enabled:true
```

7. Enable the Referential Integrity Plug-in. This will ensure that references to an entry are automatically updated when the entry is deleted or renamed.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set  
enabled:true
```

If this Directory Server is part of a replication topology, you should enable the Referential Integrity Plug-in for each replica.

Migrating Static Groups to Virtual Static Groups

In many cases, electing to use virtual static groups in place of static groups can produce marked performance gains without any need to update client applications. The specifics of a migration to virtual static groups varies depending on the original DIT, but the general approach involves identifying common membership traits for all members of each group and then expressing those traits in the form of an LDAP URL.

In the following example, the common membership trait for all members of the All Users group is the parent DN `ou=People,dc=example,dc=com`. In other cases, a common attribute may need to be used. For example, groups based on the location of its members could use the `l` (location) or `st` (state) attribute.

To Migrate DSEE Static Groups to Virtual Static Groups

In the following example, consider the common case of an "All Users" group, which contains all entries under the parent DN `"ou=People,dc=example,dc=com"`. When implemented as a virtual static group, this group may have a large membership set without incurring the overhead of a static group.

1. First, create a dynamic group.

```
dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Dynamic All Users
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(objectClass=person)
```

2. Next, create a virtual static group that references the dynamic group.

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
objectClass: ds-virtual-static-group
cn: All Users
ds-target-group-dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
```

3. Finally, the virtual Static uniqueMember virtual attribute must be enabled to populate the All Users group with uniqueMember virtual attributes.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
--set enabled:true
```

4. Confirm that the virtual static group is correctly configured by checking a user's membership in the group.

```
$ bin/ldapsearch --baseDN "cn=All Users,ou=Groups,dc=example,dc=com" \
--searchScope base "(uniqueMember=uid=user.0,ou=People,dc=example,dc=com)" 1.1
```

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
```

5. The ability to list all members of a virtual static group is disabled by default. You may enable this feature, but only if specifically required by a client application.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
--set allow-retrieving-membership: true
```



Note: The virtual static group may also be implemented using the `groupOfNames` object class instead of `groupOfUniqueNames`. In that case, you must update the Virtual Static member configuration object instead of the Virtual Static uniqueMember configuration object.

Migrating Dynamic Groups

The Ping Identity Directory Server supports dynamic groups with the `groupOfURLs` object class. In general, dynamic groups may be imported without modification.

To Migrate Dynamic Groups

1. Run the `migrate-ldap-schema` tool to enumerate any schema differences between the DSEE deployment and the PingData deployment.
2. Run the `migrate-sun-ds-config` tool to enumerate any configuration differences between the DSEE deployment and the PingData deployment.

3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the `import-ldif` tool.

Chapter

14 Encrypting Sensitive Data

The Directory Server provides several ways that you can protect sensitive information in the server. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases, and you can also protect sensitive attributes by limiting the ways that clients may interact with them.

This chapter presents the following topics:

Topics:

- [*Encrypting and Protecting Sensitive Data*](#)
- [*Backing Up and Restoring the Encryption-Settings Definitions*](#)
- [*Configuring Sensitive Attributes*](#)
- [*Configuring Global Sensitive Attributes*](#)
- [*Excluding a Global Sensitive Attribute on a Client Connection Policy*](#)

Encrypting and Protecting Sensitive Data

The Directory Server provides an encryption-settings database that holds encryption and decryption definitions to protect sensitive data. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases. You can also protect sensitive attributes by limiting the ways that clients may interact with them.

About the Encryption-Settings Database

The encryption-settings database is a repository that the server uses to hold information for encrypting and decrypting data. The database contains any number of *encryption-settings definitions* that specifies information about the cipher transformation and encapsulates the key used for encryption and decryption.

Before data encryption can be enabled, you first need to create an encryption-settings definition. An encryption-settings definition specifies the cipher transformation that should be used to encrypt the data, and encapsulates the encryption key. The `encryption-settings` command-line tool can be used to manage the encryption settings database, including creating, deleting, exporting, and importing encryption-settings definitions, listing the available definitions, and indicating which definition should be used for subsequent encryption operations.

Although the encryption-settings database can have multiple encryption-settings definitions, only one of them can be designated as the *preferred* definition. The preferred encryption-settings definition is the one that will be used for any subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten (e.g., as a result of a modify or modify DN operation, or if the data is exported to LDIF and re-imported). Similarly, if you introduce a new preferred encryption-settings definition, then any existing encrypted data will continue to use the previous definition until it is rewritten. If you do change the preferred encryption-settings definition for the server, then it is important to retain the previous definitions until you are confident that no remaining data uses those older keys.

Supported Encryption Ciphers and Transformations

The set of encryption ciphers that are supported by the Directory Server is limited to those ciphers supported by the JVM in which the server is running. For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the following:

- [Java Cryptography Architecture Reference Guide](#)
- [Java Cryptography Architecture Standard Algorithm Name Documentation](#)

When configuring encryption, the cipher to be used must be specified using a key length (in bits) and either a cipher algorithm name (e.g., "AES") or a full cipher transformation which explicitly specifies the mode and padding to use for the encryption (e.g., "AES/CBC/

PKCS5Padding"). If only a cipher algorithm is given, then the default mode and padding for that algorithm will be automatically selected.

The following cipher algorithms and key lengths have been tested using the Sun/Oracle JVM using JVM and the IBM JVM:

Table 25: Cipher Algorithms

Cipher Algorithm	Key Length (bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128



Note: By default, some JVM implementations may come with limited encryption strength, which may restrict the key lengths that can be used. For example, the Sun/Oracle JVM does not allow AES with 192-bit or 256-bit keys unless the unlimited encryption strength policy files are downloaded and installed.

The Directory Server supports four Cipher Stream Providers, which are used to obtain cipher input and output streams to read and write encrypted data.

Table 26: Cipher Stream Providers

Cipher Stream Providers	Description
Default	Default cipher stream provider using a hard-coded default key.
File-Based	Used to read a specified file in order to obtain a password used to generate cipher streams for reading and writing encrypted data.
Third-Party	Used to provide cipher stream provider implementations created in third-party code using the Server SDK.
Wait-for-Passphrase	Causes the server to wait for an administrator to enter a passphrase that will be used to derive the key for cipher streams. You can supply the passphrase to the server by running <code>encryption-settings supply-passphrase</code> .

Using the encryption-settings Tool

The `encryption-settings` tool provides a mechanism for interacting with the server's encryption-settings database. It may be used to list the available definitions, create new definitions, delete existing definitions, and indicate which definition should be the preferred definition. It may also be used to export definitions to a file for backup purposes and to allow them to be imported for use in other Directory Server instances.

To List the Available Encryption Definitions

- Use the `encryption-settings` tool with the `list` subcommand to display the set of available encryption settings definitions. This subcommand does not take any arguments. For

each definition, it will include the unique identifier for the definition, as well as the cipher transformation and key length that will be used for encryption and whether it is the preferred definition.

```
$ bin/encryption-settings list
```

```
Encryption Settings Definition ID: 4D86C7922F71BB57B8B5695D2993059A26B8FC01
Preferred for New Encryption: false
Cipher Transformation: DESede
Key Length (bits): 192

Encryption Settings Definition ID: F635E109A8549651025D01D9A6A90F7C9017C66D
Preferred for New Encryption: true
Cipher Transformation: AES
Key Length (bits): 128
```

Creating Encryption-Settings Definitions

To create a new encryption-settings definition, use the `create` subcommand. This subcommand takes the following arguments:

- **--cipher-algorithm {algorithm}**. Specifies the base cipher algorithm that should be used. This should just be the name of the algorithm (e.g., "AES", "DES", "DESede", "Blowfish", "RC4", etc.). This argument is required.
- **--cipher-transformation {transformation}**. Specifies the full cipher transformation that should be used, including the cipher mode and padding algorithms (e.g., "AES/CBC/PKCS5Padding"). This argument is optional, and if it is not provided, then the JVM-default transformation will be used for the specified cipher algorithm.
- **--key-length-bits {length}**. Specifies the length of the encryption key in bits (e.g., 128). This argument is required.
- **--set-preferred**. Indicates that the new encryption-settings definition should be made the preferred definition and therefore should be used for subsequent encryption operations in the server. When creating the first definition in the encryption-settings database, it will automatically be made the preferred definition.

To Create an Encryption-Settings Definition

- Use the `encryption-settings` tool with the `create` subcommand to specify the definition.

```
$ bin/encryption-settings create --cipher-algorithm AES \
  --key-length-bits 128 --set-preferred
```

```
Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the Preferred Encryption-Settings Definition

To change the preferred encryption-settings definition, use the `encryption-settings` tool with the `set-preferred` subcommand. This subcommand takes the following arguments:

- `--id {id}`. Specifies the ID for the encryption-settings definition to be exported. This argument is required.

To Change the Preferred Encryption-Settings Definition

- Use the `encryption-settings` tool with the `set-preferred` subcommand to change a definition to a preferred definition.

```
$ bin/encryption-settings set-preferred --id 4D86C7922F71BB57B8B5695D2993059A26B8FC01
```

```
Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
successfully set as the preferred definition for subsequent encryption operations
```

Deleting an Encryption-Settings Definition

To delete an encryption-settings definition, use the `encryption-settings` tool with the `delete` subcommand. The subcommand takes the following arguments:

- `--id {id}`. Specifies the ID for the encryption-settings definition to be deleted. This argument is required.

Never delete an encryption-settings definition if data in the server is still encrypted using the settings contained in that definition. Any data still encrypted with a definition that has been removed from the database will be inaccessible to the server and will cause errors for any attempt to access it. This includes the replicationChanges and changelog databases, which the `re-encode-entries` tool will not re-encode with the new encryption-settings definition. Therefore, wait for the amount of time defined in the `replication-purge-delay`, of the Replication Server, and `changelog-maximum-age` of the changelog Backend (if enabled) before removing previous encryption-settings definitions. To safely delete a compromised encryption-settings definition, see the [Dealing with a Compromised EncryptionKey](#) section.

To stop using a definition for encryption and use a different definition, make sure that the desired definition exists in the encryption-settings database and set it to be the preferred definition. As long as the encryption key has not been compromised, there is no harm in having old encryption-settings definitions available to the server, and it is recommended that they be retained just in case they are referenced by something.

The preferred encryption-settings definition cannot be deleted unless it is the only one left. To delete the currently-preferred definition when one or more other definitions are available, make one of the other definitions preferred as described in the previous section.

To Delete an Encryption-Settings Definition

- Use the `encryption-settings` command with the `delete` subcommand. Make sure to include the `--id` argument to specify the definition.

```
$ bin/encryption-settings delete --id F635E109A8549651025D01D9A6A90F7C9017C66D
```

```
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Configuring the Encryption-Settings Database

Because the encryption-settings database contains the encryption keys used to protect server data, the contents of the encryption-settings database is itself encrypted. By default, the server will derive a key to use for this purpose, but it is recommended that you customize the logic used to access the encryption-settings database with a cipher stream provider. The Server SDK provides an API that can be used to create custom cipher stream provider implementations, but the server also comes with one that will obtain the key from a PIN file that you create (see the example procedure below).

To Configure the Encryption-Settings Database

1. Use `dsconfig` to configure the server so that the encryption-settings database is encrypted with a PIN contained in the file `config/encryption-settings.pin`.

```
$ bin/dsconfig create-cipher-stream-provider \  
  --provider-name "Encryption Settings PIN File" \  
  --type file-based \  
  --set enabled:true \  
  --set password-file:config/encryption-settings.pin
```

2. Use `dsconfig` to set the global configuration property for the cipher stream provider, which sets the on-disk encryption.

```
$ bin/dsconfig set-global-configuration-prop \  
  --set "encryption-settings-cipher-stream-provider:Encryption Settings PIN File"
```

3. Use the `encryption-settings` tool to create a new encryption-settings definition. This command automatically generates a new 256-bit encryption key for use with AES encryption, and mark it as the preferred definition for future encryption operations in the server. Note that this command will fail if you do not have the unlimited encryption strength policy installed as described in the previous section (if you do not have that policy installed, then you are restricted to a 128-bit key for AES encryption).

```
$ bin/encryption-settings create \  
  --cipher-algorithm AES \  
  --key-length-bits 256 \  
  --set-preferred
```

4. Obtain a list of the definitions in the encryption-settings database.

```
$ bin/encryption-settings list
```

5. You can export an encryption-settings definition from the database using a command like the following where the encryption-settings ID should be changed as necessary to suit your deployment:

```
$ bin/encryption-settings export \  
  --id DA39A3EE5E6B4B0D3255BF95601890AFD80709 \  
  --output-file /tmp/exported-key \  
  --pin-file /tmp/exported-key.pin
```

6. If no PIN file is specified, then you will be interactively prompted to provide it. To import an encryption-settings definition into the database on another server.

```
$ bin/encryption-settings import \
  --input-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin \
  --set-preferred
```

Backing Up and Restoring the Encryption-Settings Definitions

If using data encryption in a Directory Server instance, do not lose the encryption-settings definitions used to encrypt data in the server. If an encryption-settings definition is lost, any data encrypted with that definition will be completely inaccessible. Make sure the encryption-settings definitions are backed up regularly.

The Directory Server provides two different mechanisms for backing up and restoring encryption-settings definitions. Individual encryption-settings definitions can be exported and imported using the `encryption-settings` tool. Or, the entire encryption-settings database can be backed up and restored using the Directory Server's `backup` and `restore` tools.

If a pin file is used to define a passphrase to the encryption-settings database, the passphrase must be backed up and kept secure independently of the userRoot and encryption-settings database backups. The passphrase in the pin file is needed if the encryption-settings database is to be restored into a different server root.

Exporting Encryption-Settings Definitions

To back up an individual definition (or to export it from one server so that you can import it into another), use the `export` subcommand to the `encryption-settings` command. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.
- **--output-file {path}**. Specifies the path to the output file to which the encryption-settings definition will be written. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.

To Export an Encryption-Settings Definition

- Use the `encryption-settings` tool with the `export` subcommand to export the definition to a file.

```
$ bin/encryption-settings export --id F635E109A8549651025D01D9A6A90F7C9017C66D \
  --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
```

```
Re-enter the encryption PIN:
```

```
Successfully exported encryption settings definition  
F635E109A8549651025D01D9A6A90F7C9017C66D to file /tmp/exported-key
```

Importing Encryption-Settings Definitions

To import an encryption-settings definition that has been previously exported, use the `encryption-settings` tool with the `import` subcommand. The subcommand takes the following arguments:

- **--input-file {path}**. Specifies the path to the file containing the exported encryption-settings definition. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.
- **--set-preferred**. Specifies that the newly-imported encryption-settings definition should be made for the preferred definition for subsequent encryption-settings.

To Import an Encryption-Settings Definition

- Use the `encryption-settings` tool with the `import` subcommand to import the definition to a file.

```
$ bin/encryption-settings import --input-file /tmp/exported-key --set-preferred  
Enter the PIN used to encrypt the definition:
```

```
Successfully imported encryption settings definition  
F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
```

Enabling Data Encryption in the Server

To enable data encryption in the server, you must have at least one encryption-settings definition available for use. Then, it is only necessary to set the value of the `encrypt-data` global configuration property to true.

Setting the global configuration property will automatically enable data encryption for all types of backends that support it (including the changelog backend) as well as for the replication server database. All subsequent write operations will cause the corresponding records written into any of these locations to be encrypted. Any existing data will remain unencrypted until it is rewritten by a write operation. If you wish to have existing data encrypted, then you will need to export that data to LDIF and re-import it. This will work for both the data backends and the changelog, but it is not an option for the replication database, so existing change records will remain unencrypted until they are purged. If this is not considered acceptable in your environment, then follow the steps in the [Dealing with a Compromised Encryption Key](#) to safely purge the replication database.

To Enable Data Encryption in the Server

- Use `dsconfig` to set the global configuration property for data encryption to true.

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using Data Encryption in a Replicated Environment

Data encryption is only used for the on-disk storage for data within the server. Whenever clients access that data, it is presented in unencrypted form (although the communication with those clients may itself be encrypted using SSL or StartTLS). Replication, the communication of updates between replication servers, is always encrypted using SSL. Each server may apply data encryption in a completely independent manner and have different sets of encryption-settings definitions. It is also possible to have a replication topology containing some servers with data encryption enabled and others with it disabled.

However, when initializing the backend of one server from another server with data encryption enabled, then the server being initialized must have access to all encryption-settings definitions that may have been used for data contained in that backend. To do this, perform an export of the encryption-settings database on the source server using `bin/encryption-settings export` and import it on the target server using `bin/encryption-settings import`.

Dealing with a Compromised Encryption Key

If an encryption-settings definition becomes compromised such that an unauthorized individual obtains access to the encryption key, then any data encrypted with that definition is also vulnerable because it can be decrypted using that key. It is very important that the encryption-settings database be protected (e.g., using file permissions and/or filesystem ACLs) to ensure that its contents remain secure.

In the event that an encryption-settings definition is compromised, then you should immediately stop using that definition. Any data encrypted with the compromised key should be re-encrypted with a new definition or purged from the server. This can be done on one server at a time to avoid an environment-wide downtime, but it should be completed as quickly as possible on all servers that had used that definition at any point in the past in order to minimize the risk of that data becoming exposed.

To Deal with a Compromised Encryption Key

The recommended process for responding to a compromised encryption settings definition is as follows:

1. Create a new encryption-settings definition and make it the preferred definition for new writes.
2. Ensure that client traffic is routed away from the server instance to be updated. For example, if the Directory Server is accessed through a Directory Proxy Server, then you may set the

`health-check-state` configuration property for any LDAP external server definitions that reference that server to have a value of `unavailable`.

3. Ensure that external clients are not allowed to write operations in the directory server instance. This may be accomplished by setting the `writability-mode` global configuration property to have a value of `internal-only`.
4. Wait for all outstanding local changes to be replicated out to other servers. This can be accomplished by looking at the monitor entries with the `ds-replication-server-handler-monitor-entry` object class to ensure that the value of the `update-sent` attribute is no longer increasing.
5. Stop the directory server instance.
6. Delete the replication server database by removing all files in the `changeLogDb` directory below the server root. As long as all local changes have been replicated out to other servers, this will not result in any data loss in the replication environment.
7. Export the contents of all local DB and changelog backends to LDIF. Then, re-import the data from LDIF, which will cause it to be encrypted using the new preferred encryption settings definition.
8. Export the compromised key from the encryption settings database to back it up in case it may be needed again in the future (e.g., if some remaining data was later found to have been encrypted with the key contained in that definition). Then, delete it from the encryption settings database so that it can no longer be used by that directory server instance.
9. Start the directory server instance.
10. Allow replication to bring the server back up-to-date with any changes processed while it was offline.
11. Re-allow externally-initiated write operations by changing the value of the global `writability-mode` configuration property back to `enabled`.
12. Re-configure the environment to allow client traffic to again be routed to that server instance (e.g., by changing the value of the "health-check-state" property in the corresponding LDAP external instance definitions in the Directory Proxy Server instances back to "dynamically-determined").

Configuring Sensitive Attributes

Data encryption is only applied to the on-disk storage for a Directory Server instance. It does not automatically protect information accessed or replicated between servers, although the server offers other mechanisms to provide that protection (i.e., SSL, StartTLS, SASL). Ensuring that all client communication uses either SSL or StartTLS encryption and ensuring that all replication traffic uses SSL encryption ensures that the data is protected from unauthorized individuals who may be able to eavesdrop on network communication. This communication security may be enabled independently of data encryption (although if data encryption is

enabled, then it is strongly recommended that secure communication be used to protect network access to that data).

However, for client data access, it may not be as simple as merely enabling secure communication. In some cases, it may be desirable to allow insecure access to some data. In other cases, it may be useful to have additional levels of protection in place to ensure that some attributes are even more carefully protected. These kinds of protection may be achieved using sensitive attribute definitions.

Each sensitive attribute definition contains a number of configuration properties, including:

- **attribute-type**. Specifies the set of attribute types whose values may be considered sensitive. At least one attribute type must be provided, and all specified attribute types must be defined in the server schema.
- **include-default-sensitive-operational-attributes**. Indicates whether the set of sensitive attributes should automatically be updated to include any operational attributes maintained by the Directory Server itself that may contain sensitive information. At present, this includes the `ds-sync-hist` operation attribute, which is used for data required for replication conflict resolution and may contain values from other attributes in the entry.
- **allow-in-filter**. Indicates whether sensitive attributes may be used in filters. This applies not only to the filter used in search requests, but also filters that may be used in other places, like the assertion and join request controls. The value of this property must be one of:
 - `Allow` (allow sensitive attributes to be used in filters over both secure and insecure connections)
 - `Reject` (reject any request which includes a filter targeting one or more sensitive attributes over both secure and insecure connections)
 - `Secure-only` (allow sensitive attributes to be used in filters over secure connections, but reject any such requests over insecure connections)
- **allow-in-add**. Indicates whether sensitive attributes may be included in entries created by LDAP add operations. The value of this property must be one of:
 - `Allow` (allow sensitive attributes to be included in add requests over both secure and insecure connections)
 - `Reject` (reject any add request containing sensitive attributes over both secure and insecure connections)
 - `Secure-only` (allow sensitive attributes to be included in add requests received over a secure connection, but reject any such requests over an insecure connection)
- **allow-in-compare**. Indicates whether sensitive attributes may be targeted by the assertion used in a compare operation. The value of this property must be one of:
 - `Allow` (allow sensitive attributes to be targeted by requests over both secure and insecure connections)
 - `Reject` (reject any compare request targeting a sensitive attribute over both secure and insecure connections)

- `Secure-only` (allow compare requests targeting sensitive attributes over a secure connection, but reject any such requests over an insecure connection)
- `allow-in-modify`. Indicates whether sensitive attributes may be updated using modify operations. The value of this property must be one of:
 - `Allow` (allow sensitive attributes to be modified by requests over both secure and insecure connections)
 - `Reject` (reject any modify request updating a sensitive attribute over both secure and insecure connections)
 - `Secure-only` (only modify requests updating sensitive attributes over a secure connection, but reject any such request over an insecure connection)

The `allow-in-returned-entries`, `allow-in-filter`, `allow-in-add`, `allow-in-compare`, and `allow-in-modify` properties all have default values of `secure-only`, which prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

If a client connection policy references a sensitive attribute definition, then any restrictions imposed by that definition will be enforced for any clients associated with that client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, then the server will use the most restrictive combination of all of those sets.

Note that sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and may only be used to enforce additional restrictions on clients. Sensitive attribute definitions may never be used to grant a client additional access to information that it would not have already had through other means. For example, if the `employeeSSN` attribute is declared to be a sensitive attribute and the `allow-in-returned-entries` property has a value of `Secure-only`, then the `employeeSSN` attribute will only be returned to those clients that have both been granted permission by the access control rules defined in the server and are communicating with the server over a secure connection. The `employeeSSN` attribute will be stripped out of entries returned to clients normally authorized to see it if they are using insecure connections, and it will also be stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

To Create a Sensitive Attribute

1. To create a sensitive attribute, you must first create one or more sensitive attribute definitions.

For example, to create a sensitive attribute definition that will only allow access to the `employeeSSN` attribute by clients using secure connections, the following configuration changes may be made:

```
$ bin/dsconfig create-sensitive-attribute \  
  --attribute-name "Employee Social Security Numbers" \  
  --set attribute-type:employeeSSN \  
  --set include-default-sensitive-operational-attributes:true \  
  --set allow-in-returned-entries:secure-only \  
  --set allow-in-filter:secure-only \  
  --set allow-in-add:secure-only \  
  --set allow-in-compare:secure-only \  
  \
```



```
--set allow-in-modify:secure-only
```

2. Associate those sensitive attribute definitions with the client connection policies for which you want them to be enforced.

```
$ bin/dsconfig set-client-connection-policy-prop --policy-name default \
--set "sensitive-attribute:Employee Social Security Numbers"
```

Configuring Global Sensitive Attributes

Administrators can assign one or more sensitive attribute definitions to a client connection policy. However, in an environment with multiple client connection policies, it could be easy to add a sensitive attribute definition to one policy but overlook it in another. The Directory Server supports the ability to define sensitive attributes as a global configuration option so that they will automatically be used across all client connection policies.

To Configure a Global Sensitive Attribute

- Run `dsconfig` to add a global sensitive attribute across all client connection policies. The following command adds the `employeeSSN` as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop --add "sensitive-attribute:employeeSSN"
```

Excluding a Global Sensitive Attribute on a Client Connection Policy

Administrators can set a global sensitive attribute across all client connection policies. However, there may be cases when a specific directory server must exclude the sensitive attribute as it may not be needed for client connection requests. For example, in most environments it is good to declare the `userPassword` attribute to be a sensitive attribute in a manner that prevents it from being read by external clients. Further, this solution is more secure than protecting the `password` attribute using the server's default global ACI, which only exists for backwards compatibility purposes. If the Data Sync Server is installed, then it does need to be able to access passwords for synchronization purposes. In this case, the administrator can set `userPassword` to be a sensitive attribute in all client connection policies, but exclude it in a policy specifically created for use by the Data Sync Server. The Directory Server provides an `exclude-global-sensitive-attribute` property for this purpose.

To Exclude a Global Sensitive Attribute on a Client Connection Policy

1. Run `dsconfig` to remove the global ACI that limits access to the `userPassword` or `authPassword` attribute. This is present for backwards compatibility.

```
$ bin/dsconfig set-access-control-handler-prop \
--remove 'global-aci:(targetattr="userPassword || authPassword")'
```

```
(version 3.0; acl "Prevent clients from retrieving passwords from the server";  
deny (read,search,compare) userdn="ldap:///anyone";)
```

2. Run `dsconfig` to add the `userPassword` attribute as a global sensitive attribute, which is applied to all client connection policies. Do this by adding the built-in "Sensitive Password Attributes" Sensitive Attribute definition to the Global Configuration.

```
$ bin/dsconfig set-global-configuration-prop \  
--add "sensitive-attribute:Sensitive Password Attributes"
```

3. If the server is designated to synchronize passwords with a Sync Server, then it is necessary to configure a client connection policy for the Sync User to exclude the global sensitive attribute. The following is an example on how to create a new policy if the Data Sync Server binds with the default DN of `cn=Sync User,cn=Root DNs,cn=config`.

```
$ bin/dsconfig create-connection-criteria \  
--criteria-name "Requests by Sync Users" \  
--type simple \  
--set user-auth-type:internal \  
--set user-auth-type:sasl \  
--set user-auth-type:simple \  
--set "included-user-base-dn:cn=Sync User,cn=Root DNs,cn=config"  
  
$ bin/dsconfig create-client-connection-policy \  
--policy-name "Data Sync Server Connection Policy" \  
--set enabled:true \  
--set evaluation-order-index:9998 \  
--set "connection-criteria:Requests by Sync Users" \  
--set "exclude-global-sensitive-attribute:Sensitive Password Attributes"
```

Chapter

15 Working with the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services.

This chapter will present the following topics related to the LDAP Changelog:

Topics:

- [*Overview of the LDAP Changelog*](#)
- [*Viewing the LDAP Changelog Properties*](#)
- [*Enabling the LDAP Changelog*](#)
- [*Changing the LDAP Changelog Database Location*](#)
- [*Viewing the LDAP Changelog Parameters in the Root DSE*](#)
- [*Viewing the LDAP Changelog Using ldapsearch*](#)
- [*Indexing the LDAP Changelog*](#)
- [*Tracking Virtual Attribute Changes in the LDAP Changelog*](#)

Overview of the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services. For example, you can install the Data Sync Server that monitors the LDAP changelog for any updates that occur on a source directory server and synchronizes these changes to a target DIT or database server. The Directory Server provides an additional feature in that the LDAP changelog supports virtual attributes.

Note: The LDAP Changelog should not be confused with the Replication Changelog. The main distinction is as follows:



- The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`.
 - The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB`.
-

Key Changelog Features

As of version 3.2, the Directory Server supports two new Changelog Backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries. External client applications can change the contents of attributes they can see in the targeted entry based on the access control rules applied to the associated base DN.

- **apply-access-controls-to-changelog-entry-contents.** Indicates whether the contents of changelog entry attributes (i.e., `changes`, `deletedEntryAttrs`, `ds-changelog-entry-key-attr-values`, `ds-changelog-before-values`, and `ds-changelog-after-values`) are subject to access control and/or sensitive attribute evaluation to limit data that LDAP clients can see. The client must have the access control permissions to read changelog entries to retrieve them in any form. If this feature is enabled and the client does not have permission to read an entry at all, or if that client does not have permission to see any attributes that were targeted by the change, then the associated changelog entries targeted by those operations will be suppressed. If a client does not have permission to see certain attributes within the target entry, then references to those attributes in the changelog entry will also be suppressed. This property only applies to standard LDAP searches of the `cn=changelog` branch.
- **report-excluded-changelog-attributes.** Indicates whether to include additional information about any attributes that may have been removed due to access control filtering. This property only applies to content removed as a result of processing performed by the `apply-access-controls-to-changelog-entry-contents` property. Possible values are:
 - **none** - Indicates that changelog entries should not include any information about attributes that have been removed.

- **attribute-counts** - Indicates that changelog entries should include a count of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes will be reported in the `ds-changelog-num-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the number of the excluded operational attributes will be reported in the `ds-changelog-num-excluded-operational-attributes` attribute of the changelog entry. Both the `ds-changelog-num-excluded-user-attributes` and `ds-changelog-num-excluded-operational-attributes` are operational and must be explicitly requested by clients (or all operational attributes requested using "+") to be returned.
- **attribute-names** - Indicates that changelog entries should include the names of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, then the names of the excluded user attributes will be reported in the `ds-changelog-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the names of the excluded operational attributes will be reported in the `ds-changelog-excluded-operational-attribute` attribute of the changelog entry. Both the `ds-changelog-excluded-user-attribute` and `ds-changelog-excluded-operational-attribute` attributes are operational and must be explicitly requested by clients (or all operational attributes requested via "+") to be returned.

To Enable Access Control Filtering in the LDAP Changelog

To set up access control to the LDAP Changelog, use the `dsconfig` tool to enable the properties to the Changelog Backend. Only admin users with the `bypass-acl` privilege can read the changelog.

1. Enable the `apply-access-control-to-changelog-entry-contents` property to allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "apply-access-controls-to-changelog-entry-contents:true"
```

Access control filtering will be applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the Changelog Backend is servicing requests from an Data Sync Server that has the `filter-changes-by-user` Sync Pipe property set.

2. Optional. Set the `report-excluded-changelog-attributes` property to include a count of users that have been removed through access control filtering. The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "report-excluded-changelog-attributes:attribute-counts"
```

Useful Changelog Features

The Directory Server provides two useful changelog configuration properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

- **changelog-max-before-after-values.** Setting this property to a non-zero value causes all of the old values and all of the new values (up to the specified maximum) for each changed attribute to be stored in the changelog entry. The values will be stored in the `ds-changelog-before-values` and `ds-changelog-after-values` attributes on the changelog entry. These attributes are not present by default.



Note: The `changelog-max-before-after-values` property can be expensive for attributes with hundreds or thousands of values, such as a group entry.

If any attribute has more than the maximum number of values, their names and number of before/after values will be stored in the `ds-changelog-attr-exceeded-max-values-count` attribute on the changelog entry. This is a multi-valued attribute whose format is:

```
attr=attributeName,beforeCount=100,afterCount=101
```

where "attributeName" is the name of the attribute and the "beforeCount" and "afterCount" are the total number of values for that attribute before and after the change, respectively. This attribute indicates that you need to reset the `changelog-max-before-after-values` property to a higher value. When this attribute is set, an alert will be generated.



Note: If the number of values for an attribute exceeds the maximum value set by the `changelog-max-before-after-values` property, then those values will not be stored.

- **changelog-include-key-attribute.** This property is used for correlation attributes that need to be synchronized across servers, such as `uid`. It causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the changelog entry. This applies for all change types. On a DELETE operation, the values are from the entry before they were deleted.

The key values will be recorded on every change and override the settings configured in `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry-exclude-attribute`.

Example of the Changelog Features

After the `changelog-max-before-after-values` property is set, the before-and-after values of any change attribute will be recorded in the LDAP Changelog. For example, given a simple entry with two multi-valued mail attributes:

After base64 decoding, the `ds-changelog-after-values` attribute reads:

```
description: newDescription
mail: test@gmail.com
mail: test@hotmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

Viewing the LDAP Changelog Properties

You can view the LDAP changelog properties by running the `dsconfig get-backend-prop` command and specifying the changelog backend.

To View the LDAP Changelog Properties Using `dsconfig` Non-Interactive Mode

- Use `dsconfig` to view the changelog properties on the Directory Server. To view "advanced" properties that are normally hidden, add the `--advanced` option when running the command. For a specific description of each property, see the *Ping Identity Directory Server Configuration Reference*.

```
$ bin/dsconfig get-backend-prop --backend-name changelog
```

```
Property                               : Value(s)
-----
backend-id                             : changelog
description                             : -
enabled                                 : false
writability-mode                        : disabled
base-dn                                 : cn=changelog
set-degraded-alert-when-disabled        : false
return-unavailable-when-disabled        : false
db-directory                            : db
db-directory-permissions                 : 700
changelog-maximum-age                   : 2 d
db-cache-percent                         : 1
changelog-include-attribute              : -
changelog-exclude-attribute              : -
changelog-deleted-entry-include-attribute : -
changelog-deleted-entry-exclude-attribute : -
changelog-include-key-attribute          : -
changelog-max-before-after-values       : 0
changelog-write-batch-size               : 100
changelog-purge-batch-size               : 1000
changelog-write-queue-capacity           : 100
write-lastmod-attributes                 : true
use-reversible-form                      : false
je-property                              : -
```

Enabling the LDAP Changelog

By default, the LDAP changelog is disabled on the Directory Server. If you are using the `dsconfig` tool in interactive mode, the changelog appears in the Backend configuration as a Standard object menu item.



Note: You can enable the feature using the `dsconfig` tool only if required as it can significantly affect LDAP update performance.

To Enable the LDAP Changelog Using `dsconfig` Non-Interactive Mode

- Use `dsconfig` to enable the changelog property on the Directory Server.

```
$ bin/dsconfig set-backend-prop \
  --backend-name changelog --set enabled:true
```

To Enable the LDAP Changelog Using Interactive Mode

1. Use `dsconfig` to enable the changelog on each server in the network. Then, authenticate to the server by entering the host name, LDAP connection, port, bindDN and bind password.

```
$ bin/dsconfig
```

2. On the Directory Server main menu, type `o` to change from the Basic object level to the Standard object level.
3. Enter the option to select the Standard Object level.
4. On the Directory Server main menu, type the number corresponding to Backend.
5. On the **Backend Management** menu, enter the option to view and edit an existing backend.
6. Next, you should see a list of the accessible backends on your system. For example, you may see options for the `changelog` and `userRoot` backends. Enter the option to work with the `changelog` backend.
7. On the **Changelog Backend properties** menu, type the number corresponding to the Enabled property.
8. On the **Enabled Property** menu, type the number to change the `Enabled` property to `TRUE`.
9. On the **Backend Properties** menu, type `f` to apply the change. If you set up the server in a server group, type `g` to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.
10. Verify that changes made to the data are recorded in the changelog.

Changing the LDAP Changelog Database Location

In cases where disk space issues arise, you can change the on-disk location of the LDAP Change Log database. The changelog backend supports a `db-directory` property that specifies the absolute or relative path (i.e., relative to the local server root) to the filesystem directory that

is used to hold the Oracle Berkeley DB Java Edition database files containing the data for this backend.

If you change the changelog database location, you must stop and then restart the Directory Server for the change to take effect. If the changelog backend is already enabled, then the database files must be manually moved or copied to the new location while the server is stopped.

To Change the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. Use `dsconfig` to change the database location for the LDAP Changelog, which by default is at `<server-root>/db`. The following command sets the LDAP changelog backend to `<server-root>/db2`. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \  
--set "db-directory:db2" --set "enabled:true"
```

The database files are stored under `<server-root>/db2/changelog`. The files for this backend are stored in a sub-directory named after the `backend-id` property.

2. Stop and restart the server. Since the LDAP changelog backend was previous disabled, there is no need to manually relocate any existing database files.

```
$ bin/stop-ds  
$ bin/start-ds
```

To Reset the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. If you have changed the LDAP Changelog location, but want to reset it to its original location, use `dsconfig` to reset it. The following command resets the LDAP changelog backend to `<server-root>/db` location. Remember to include the LDAP connection parameters (e.g., hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \  
--reset "db-directory"
```

2. The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it will create an empty database. If the LDAP changelog backend at the previous location is enabled at the time, stop the server, manually copy the database files to the new LDAP changelog location, and then restart the server.

Viewing the LDAP Changelog Parameters in the Root DSE

The Root DSE is a special entry that holds operational information about the server. The entry provides information about the LDAP controls, extended operations, and SASL mechanisms available in the server as well as the state of the data within the changelog. For changelog parameters, the attributes of interest include:

- **firstChangeNumber.** Change number for the first (oldest) change record contained in the LDAP changelog.
- **lastChangeNumber.** Change number for the last (most recent) change record contained in the LDAP changelog.
- **lastPurgedChangeNumber.** Change number for the last change that was purged from the LDAP changelog. It can be 0 if no changes have yet been purged.
- **firstReplicaChange.** Information about the first (oldest) change record for a change received from the specified replica. This is a multi-valued attribute and should include a value for each server in the replication topology.
- **lastReplicaChange.** Information about the last (most recent) change record for a change received from the specified replica.

The `firstReplicaChange` and `lastReplicaChange` attributes use the following syntax:

```
serverID:CSN:changeNumber
```

where:

- **serverID.** Specifies the unique identifier for the server updating the change log.
- **CSN.** Specifies the Change Sequence Number, which is the time when the update was made to the given replica.
- **changeNumber.** Specifies the order of the change that is logged to the LDAP changelog.

The `firstReplicaChange` and `lastReplicaChange` attributes can be used to correlate information in the local LDAP Change Log with data in the LDAP Change Log of other servers in the replication topology. The order of the individual changes in the LDAP Change Log can vary between servers based on the order in which they were received from a replica.

To View the LDAP Changelog Parameters

- Use `ldapsearch` to view the Root DSE.

```
$ bin/ldapsearch --baseDN "" --searchScope base "(objectclass=*)" "+"
```

Viewing the LDAP Changelog Using `ldapsearch`

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. You can view changelog entries using `ldapsearch`. Changes are represented in the form documented in the *draft-good-ldap-changelog* specification with the `targetDN` attribute providing the DN of the updated entry, the `changeType` attribute providing the type of operation (add, delete, modify, or modDN), and the `changes` attribute providing a base64-encoded representation of the attributes included in the entry (for add operations) or the changes made (for modify operations) in LDIF form. You can view the changes by decoding the encoded value using the `base64 decode` utility. The LDAP SDK for Java also provides support for parsing changelog entries.

To View the LDAP Changelog Using ldapsearch

1. Grant access to the `cn=changelog` backend to the `uid=admin` account using access control rules. By default, only the root user has access to this backend.

```
$ bin/ldapmodify
dn: cn=changelog
changetype: modify
add: aci
aci: (targetattr="*|+")(
  (version 3.0; acl "Access to the changelog backend for the admin account";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

2. Use `ldapsearch` to view the changelog.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)"

dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.0,ou=People,dc=example,dc=com
changeType: modify
changes:: cmVwbGFjZTogbW9iaWxlCm1vYmlsZTogKzEgMDIwIDE1NCA5Mzk4Ci0KcmVwbGFjZTogaG9tZVBob25lCmhvbWVQaG9uZTogKzEgMjI1IDlxNiA0OTQ5Ci0KcmVwbGFjZTogZ2l2ZW50YW1lCmdpdmVuTmFtZTogQWYyb24KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbGpkaXNjcmlwdGlvbjoGdGhpcyBpcyB0aGUgZGVzY3JpcHRpb24gZm9yIEFhcm9uIEF0cC4KLQpyZXBsYWNlOiBtb2RpbmllcnNOYW1lCm1vZGlmYWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VvLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFBQkhQOHpUR0E9Cgo=
changenumber: 1

dn: changeNumber=2,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: dc=example,dc=com
changeType: modify
changes:: cmVwbGFjZTogZHMtc3luYy1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0QzYwNDA5MzAwMDAwMDAyCgo=
changenumber: 2
```

To View the LDAP Change Sequence Numbers

- The changelog displays the server state information, which is important for failover between servers during synchronization operations. The server state information is exchanged between the servers in the network (LDAP servers and replication servers) as part of the protocol start message. It also helps the client application determine which server is most up-to-date. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"

dn: cn=changelog

dn: changeNumber=1,cn=changelog
entry-size-bytes: 182
targetUniqueId: 68147342-1f61-3465-8489-3de58c532130
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184D9E303000000001
replicationCSN: 0000011D27184D9E303000000001
replicaIdentifier: 12336
```

```
dn: changeNumber=2,cn=changelog
entry-size-bytes: 263
targetUniqueId: 4e9b7847-edcb-3791-b11b-7505f4a55af4
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184F2E303000000002
replicationCSN: 0000011D27184F2E303000000002
replicaIdentifier: 12336
```

To View LDAP Changelog Monitoring Information

The changelog contains a monitor entry that you can access over LDAP, JConsole, the Administrative Console, or SNMP. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

- Use `ldapsearch` to view the changelog monitor entry.

```
$ bin/ldapsearch --baseDN cn=changelog,cn=monitor "(objectclass=*)"
```

```
dn: cn=changelog,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
cn: changelog
changelog: cn=changelog
firstchangenumber: 1
lastchangenumber: 8
lastpurgedchangenumber: 0
firstReplicaChange: 16225:0000011D0205237F3F6100000001:5
firstReplicaChange: 16531:0000011CFF334C60409300000002:1
lastReplicaChange: 16225:0000011D02054E8B3F6100000002:7
lastReplicaChange: 16531:0000011CFF334C60409300000002:1
oldest-change-time: 20081015063104Z
...(more data)...
```

Indexing the LDAP Changelog

The Directory Server supports attribute indexing in the Changelog Backend to allow Get Changelog Batch requests to filter results that include only changes involving specific attributes. Normally, the directory server that receives a request must iterate over the whole range of changelog entries and then match entries based on search criteria for inclusion in the batch. The majority of this processing also involves determining whether the changelog entry includes changes to a particular attribute or set of attributes, or not. Using changelog indexing, client applications can dramatically speed up throughput when targeting the specific attributes.

Administrators can configure attribute indexing using the `index-include-attribute` and `index-exclude-attribute` properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values `"*"` to specify all user attributes or `"+"` to specify all operational attributes.

To determine if the directory server supports this feature, administrators can view the Root DSE for the following entry:

```
supportedFeatures: 1.3.6.1.4.1.30221.2.12.3
```

To Index a Changelog Attribute

1. Use `dsconfig` to set attribute indexing on an attribute in the Changelog Backend.

The following command enables the Changelog Backend and sets the backend to include all user attributes ("`*`") for ADD or MODIFY operations using the `changelog-include-attribute` property. The `changelog-deleted-entry-include-attribute` property is set to all attributes ("`*`") to specify a set of attribute types that should be included in a changelog entry for DELETE operations. Attributes specified in this list will be recorded in the `deletedEntryAttrs` attribute on the changelog entry when an entry is deleted. The attributes `displayName` and `employeeNumber` are indexed using the `index-include-attribute` property.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \  
--set "enabled:true" \  
--set "changelog-include-attribute:*" \  
--set "changelog-deleted-entry-include-attribute:*" \  
--set "index-include-attribute:displayName" \  
--set "index-include-attribute:employeeNumber"
```

2. Add another attribute to index using the `dsconfig --add` option, which adds the attribute to an existing configuration setting.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \  
--add "index-include-attribute:cn"
```

To Exclude Attributes from Indexing

- Use `dsconfig` to set attribute indexing on all user attributes in the Changelog Backend. The following command includes all user attributes except the description and location attributes.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \  
--set "index-include-attribute:*" \  
--set "index-exclude-attribute:description" \  
--set "index-exclude-attribute:location"
```

Tracking Virtual Attribute Changes in the LDAP Changelog

By default, the LDAP Changelog tracks changes to real attributes only. For client applications that require change tracking to include virtual attributes, administrators can enable the `include-virtual-attribute` property, so that real and virtual attributes are tracked within the changelog. Once the `include-virtual-attribute` property is enabled, then properties for virtual attributes that store before/after values, key attributes, and added or deleted entry attributes can be enabled.

To Track Virtual Attribute Changes in the LDAP Changelog

- Use `dsconfig` to enable virtual attribute change tracking in the LDAP Changelog.

The following command enables the LDAP changelog and sets `include-virtual-attributes` to `add-attributes`, which indicates that virtual attribute be included in the set of attributes listed for an add operation. The `delete-entry-attributes` option indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The `before-and-after-values` option indicates that virtual attributes should be included in the set of before and after values for attributes targeted by the changes. The `key-attribute-values` option indicates that virtual attributes should be included in the set of entry key attribute values.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \  
  --set "enabled:true" \  
  --set "include-virtual-attributes:add-attributes" \  
  --set "include-virtual-attributes:deleted-entry-attributes" \  
  --set "include-virtual-attributes: before-and-after-values" \  
  --set "include-virtual-attributes: key-attribute-values"
```


Chapter

16 Managing Access Control

The Ping Identity Directory Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and provides examples that illustrate the use of key access control functionality.

Topics:

- [*Overview of Access Control*](#)
- [*Working with Targets*](#)
- [*Examples of Common Access Control Rules*](#)
- [*Validating ACIs Before Migrating Data*](#)
- [*Migrating ACIs from Sun/Oracle to Ping Identity Directory Server*](#)
- [*Working with Privileges*](#)
- [*Working with Proxied Authorization*](#)

Overview of Access Control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources.** *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name.** *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients.** *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights.** *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The Ping Identity Directory Server provides important access control features that provide added security for the Directory Server's entries.

Improved Validation and Security

The Directory Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to

an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the Ping Identity Directory Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like `dsconfig` and the Administrative Console.

The global ACIs available by default in the Ping Identity Directory Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.
- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

Access Controls for Public or Private Backends

The Ping Identity Directory Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The Ping Identity Directory Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissible operations. When a client application, such as `ldapsearch`, enters a request, the Directory Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.

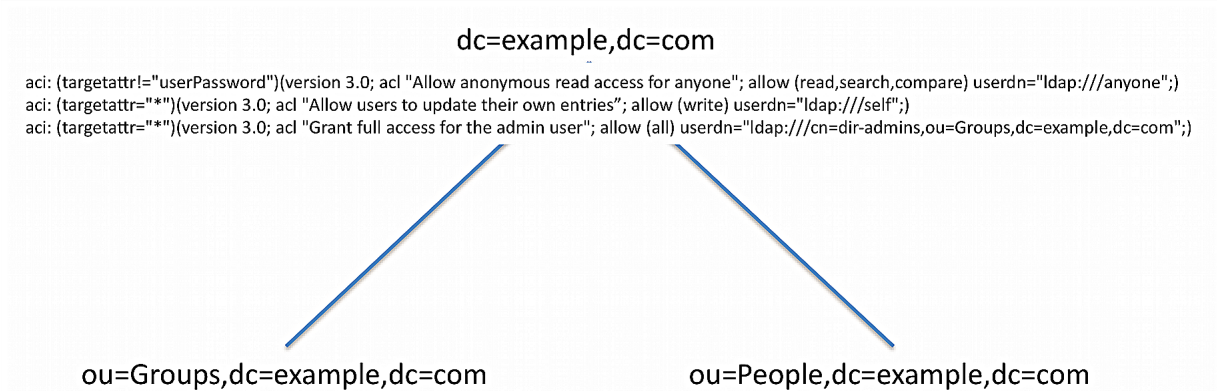


Figure 6: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 27: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: <i>(target keyword = != expression)</i>
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: <i>allow deny (permission)</i>
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: <i>bind rule keyword = != expression;</i> . The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the Ping Identity Directory Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targetattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section [Working with Targets](#).

```
(keyword [=|!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 28: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	Yes
targetattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	Yes
targetattr	Specifies the set of attributes to which the access control rule should apply.	Yes
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	Yes
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI if there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 29: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.

Permission	Description
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the <code>export</code> permission for operations against the entry's original DN. The requestor must have the <code>import</code> permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the <code>write</code> permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the <code>export</code> permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identity to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except "proxy." This is equivalent to providing a permission of "add, compare, delete, read, search, selfwrite, write."

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=|!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 30: Summary of Bind Rule Keywords

Bind Rule Keyword	Description
authmethod	<p>Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:</p> <pre>authmethod = method</pre> <p>where <i>method</i> is one of the following representations:</p> <ul style="list-style-type: none"> > none > simple. Indicates that the client is authenticated to the server using a bind DN and password. > ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server. > sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism. <p>The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (userdn="ldap:///self" and authmethod="ssl");)</pre>
dayofweek	<p>Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:</p> <pre>dayofweek = day1, day2, ...</pre> <p>where <i>day</i> is one of the following representations:</p> <ul style="list-style-type: none"> > sun > mon > tues > wed > thu > fri > sat <p>The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (dayofweek!="sun,sat" and userdn="ldap:///self" and authmethod="ssl");)</pre>
dns	<p>Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:</p> <pre>dns = dns-host-name</pre> <p>The following example allows users on hostname <code>server.example.com</code> to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (dns="server.example.com" and userdn="ldap:///self");)</pre>

Bind Rule Keyword	Description
groupdn	<p>Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.</p> <pre>groupdn [= !=] "ldap:///groupdn [ldap:///groupdn] ..."</pre> <p>The following example allows users in the managers group to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (groupdn="ldap:///cn=managers,ou=groups,dc=example,dc=com");)</pre>
ip	<p>Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:</p> <pre>ip [= !=] ipAddressList</pre> <p>where <i>ipAddressList</i> is one of the following representations:</p> <ul style="list-style-type: none"> > A specific IPv4 address: 127.0.0.1 > An IPv4 address with wildcards to specify a subnetwork: 127.0.0.* > An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0 > An IPv4 address range using CIDR notation: 123.4.5.0/24 > An IPv6 address as defined by RFC 2373. <p>The following example allows users on 10.130.10.2 and localhost to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) (ip="10.130.10.2,127.0.0.1" and userdn="ldap:///self");)</pre>
timeofday	<p>Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:</p> <pre>timeofday [= != >= > <= <] time</pre> <p>where <i>time</i> is one of the following representations:</p> <ul style="list-style-type: none"> > 4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour) > Wildcards are not allowed in this expression <p>The following example allows users to update their own entries if the request is received before 12 noon.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users who authenticate before noon to update their own entries"; allow (write) (timeofday<1200 and userdn="ldap:///self" and authmethod="simple");)</pre>
userattr	<p>Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of USERDN indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of GROUPDN indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of LDAPURL indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value</p>

Bind Rule Keyword	Description
	<p>other than USERDN, GROUPODN, or LDAPURL is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:</p> <pre>userattr = attrName# [bindType attrValue]</pre> <p>where:</p> <ul style="list-style-type: none"> > <i>attrName</i> = name of the attribute for matching > <i>bindType</i> = USERDN, GROUPODN, LDAPURL > <i>attrValue</i> = an attribute value. Note that the <i>attrVALUE</i> of the attribute must match on both the bind entry and the target of the ACI. <p>The following example allows a manager to change employee's entries. If the bind DN is specified in the <i>manager</i> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow a manager to change employee entries"; allow (write) userattr="manager#USERDN");</pre> <p>The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the <i>allowEditors</i> attribute of the targeted entry, the bind rule is evaluated to TRUE.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow allowEditors to change employee entries"; allow (write) userattr="allowEditors#GROUPODN");</pre> <p>The following example allows allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow managers to change employees entries two levels below"; allow (write) userattr="parent[0,1,2].manager#USERDN");</pre> <p>The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow any member of Eng Dept to update any other member of the engineering department at or below the ACI"; allow (write) userattr="department#ENGINEERING");</pre> <p>The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the <i>allowedEditorCriteria</i> attribute of the target entry.</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow a user that matches the filter to change entries"; allow (write) userattr="allowedEditorCriteria#LDAPURL");</pre>
userdn	<p>Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:</p> <pre>userdn [= !=] "ldap:///value ["ldap:///value ..."]</pre> <p>where <i>value</i> is one of the following representations:</p> <ul style="list-style-type: none"> > The DN of the target user > A value of <i>anyone</i> to match any client, including unauthenticated clients. > A value of <i>all</i> to match any authenticated client.

Bind Rule Keyword	Description
	<p>> A value of <code>parent</code> to match the client authenticated as the user defined in the immediate parent of the target entry.</p> <p>> A value of <code>self</code> to match the client authenticated as the user defined in the target entry.</p> <p>If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.</p> <p>The following example allows users to update their own entries:</p> <pre>aci: (targetattr="*") (version 3.0; acl "Allow users to update their own entries"; allow (write) userdn="ldap:///self");</pre>

Working with Targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///DN)
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (`!=`) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following

target expression matches all entries that contains and begins with "john.d," so that entries like "john.doe,ou=People,dc=example,dc=com," and "john.davies,ou=People,dc=example,dc=com" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the ou attribute. Entries like "john.doe,ou=People,dc=example,dc=com," and "john.davies,ou=asia-branch,dc=example,dc=com" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the ou=People,dc=example,dc=com branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
(targetattr="*")
(version 3.0; acl "Allow all the ou=People branch to modify their own entries";
allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the Ping Identity Directory Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3|...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.
- **(targetattr!="attr1||attr2||attr3|...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overridden default constraints like size limit, time limit, or look-through limit for that user.

The Ping Identity Directory Server distinguishes between these two types of attributes in its access control implementation. The Directory Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
  read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone");
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
  allow (write) userdn="ldap:///self");
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com");
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
  allow (write) userdn="ldap:///self");
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full

rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the Ping Identity Directory Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter = ldap_filter)
```

For example, the following clause targets all entries that contain "ou=engineering" attribute:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
(targetattr="departmentNumber||cn||sn")
(version 3.0; acl "example"; allow (write)
 userdn="ldap:///uid=eng-mgr,dc=example,dc=com");
```

targetfilters

The `targetfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targetfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

> **add** represents the operation of adding an attribute value to the entry

- > **del** represents the operation of removing an attribute value from the entry
- > **attr1, attr2... attrN** represents the targeted attributes
- > **filter1, filter2 ... filterN** represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.
- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add-ds-privilege-name:(ds-privilege-name=soft-delete-read)")
(version 3.0; acl "Allow members of the directory server admins group to grant the
soft-delete-read privilege"; allow (write)
groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is `base` to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
(targetattr="supportedControl||supportedExtension||
supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
(version 3.0; acl "Allow users to view Root DSE Operational Attributes";)
```

```
allow (read,search,compare) userdn="ldap:///anyone"
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
(version 3.0; acl "Allow admins to use the Soft Delete Request Control,
Hard Delete Request Control,Undelete Request Control, and
Soft-deleted entry access request control";
allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
(version 3.0; acl "Allows the mgr to use the Password Modify Request and StartTLS;
allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator Access

The following ACI can be used to grant any member of the `"cn=admins,ou=groups,dc=example,dc=com"` group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and `password policy state`:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or compare
operational attributes";
allow (read,search,compare) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

```
aci: (targetattr="*)(version 3.0; acl "Administrators can add, modify and delete
entries";
allow (all) groupdn="ldap:///cn=admin,ou=groups,dc=example,dc=com");
```

Anonymous and Authenticated Access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries representing
people";
allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries representing
people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can change
password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to `false`.

Delegated Access to a Manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy Authorization

The following ACIs can be used to allow the application `"cn=OnBehalf,ou=applications,dc=example,dc=com"` to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com");
```


Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the Ping Identity Directory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a Ping Identity Directory Server.

To validate an access control instruction, the Ping Identity Directory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

The `validate-acis` tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the `bin/validate-acis` tool (UNIX or Linux systems) or `bat\validate-acis` (Windows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
    (version 3.0; acl "Allow anonymous read access for anyone";
      allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
    (version 3.0; acl "Allow users to update their own entries";
      allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
```

```
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
   allow (all) userdn="ldap:///uid=admin,dc=example,dc=com");
```

3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif \
  --onlyReportErrors
```

```
# Processing complete
# Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

The output file shows the following message:

```
# The following access control rule is malformed or contains an unsupported
# syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
# "Allow anonymous read access for anyone"; allow (read,search,compare)
# userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
# Instruction (ACI) because it failed general ACI syntax evaluation
dn: dc=example,dc=com
aci: (targetattr!=userPassword)
  (version 3.0; acl "Allow anonymous read access for anyone";
   allow (read,search,compare) userdn="ldap:///anyone");

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
   allow (write) userdn="ldap:///self");

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
   allow (all) userdn="ldap:///uid=admin,dc=example,dc=com");
```

To Validate ACIs in Another Directory Server

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another Directory Server that you are planning to migrate to the Ping Identity Directory Server. The tool helps to determine whether the PingData Server accepts those ACIs.

- To use it in this manner, provide arguments that specify the address and port of the target Directory Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to Ping Identity Directory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the Ping Identity Directory Server.

Support for Macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the Ping Identity Directory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The Ping Identity Directory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the Ping Identity Directory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except `userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName|modifiersName"`. All operational attributes can be targeted using the "+" character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*|+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and Ping Identity Directory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the Ping Identity Directory Server, global ACIs are managed with `dsconfig` referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the Ping Identity Directory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as `"oid={oid},cn=features,cn=config"` where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: `"oid=2.16.840.1.113730.3.4.3,cn=features,cn=config"`.

In the Ping Identity Directory Server, the `"targetcontrol"` keyword can be used to define ACIs that grant or deny access to controls. The `"extop"` keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The Ping Identity Directory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.



Note: Consider running the `import-ldif` tool with the `--rejectFile` option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The Ping Identity Directory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The Ping Identity Directory Server provides a `validate-acis` tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the Ping Identity Directory Server instance. When migrating data from a DSEE deployment into an Ping Identity Directory Server instance, the `validate-acis` tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the Ping Identity Directory Server.

For more information about using this tool, see [Validating ACIs Before Migrating Data](#).

Working with Privileges

In addition to the access control implementation, the Ping Identity Directory Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the Ping Identity Directory Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the Ping Identity Directory Server.

Table 31: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the <code>audit-data-security</code> tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the <code>bypass-acl</code> privilege but without the <code>config-read</code> privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the <code>config-read</code> privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Server. The user is also required to have the <code>jmx-read</code> privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Server via the Java Management Extensions (JMX). The user is also required to have

Privilege	Description
	the <code>jmx-read</code> privilege. Note that currently all of the information exposed by the server over JMX is read-only.
<code>ldif-export</code>	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
<code>ldif-import</code>	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the <code>ldif-export</code> privilege.
<code>lockdown-mode</code>	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
<code>modify-aci</code>	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the <code>aci</code> operational attribute.
<code>password-reset</code>	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the <code>userPassword</code> attribute to the target entry.
<code>privilege-change</code>	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the <code>ds-privilege-name</code> operational attribute.
<code>proxied-auth</code>	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
<code>server-restart</code>	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
<code>server-shutdown</code>	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
<code>soft-delete-read</code>	This privilege is required for a user to access a soft-deleted-entry.
<code>stream-values</code>	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
<code>unindexed-search</code>	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
<code>update-schema</code>	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of `FALSE` in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

- > audit-data-security
- > backend-backup
- > backend-restore
- > bypass-acl
- > config-read
- > config-write
- > disconnect-client
- > ldif-export
- > lockdown-mode
- > manage-topology
- > metrics-read
- > modify-acl
- > password-reset
- > permit-get-password-policy-state-issues
- > privilege-change
- > server-restart
- > server-shutdown
- > soft-delete-read
- > stream-values
- > unindexed-search
- > update-schema

The privileges not granted to root users by default includes:

- > bypass-pw-policy
- > bypass-read-acl
- > jmx-read
- > jmx-write
- > jmx-notify
- > permit-externally-processed-authentication

- > permit-proxied-mschapv2-details
- > proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Additional Privileges for Administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
--add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*|+")(
(version 5.0; acl "Access to the tasks backend for administrators";
allow (all) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com"); )'
```

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the Ping Identity Directory Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Working with Proxied Authorization

The Directory Server supports the Proxied Authorization Control (RFC 4370) to allow an authorized LDAP client to authenticate to the server as another user. Typically, LDAP servers are deployed as backend authentication systems that store user credentials and authorization privileges necessary to carry out an operation. Single sign-on (SSO) systems can retrieve user credentials from the Directory Server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The use of the proxied authorization control provides a means for client applications to securely process requests without the need to bind or re-authenticate to the server for each and every operation.

The Directory Server supports the proxied authorization V1 and V2 request controls. The proxied authorization V1 request control is based on early versions of the draft-weltman-ldapv3-proxy Internet draft and is available primarily for legacy systems. It is recommended that deployments use the proxied authorization V2 request control based on RFC 4370.

The proxied authorization V2 control is used to request that the associated operation be performed as if it has been requested by some other user. This control may be used in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In that case, the associated operation will be processed under the authority of the specified authorization identity rather than the identity associated with the client connection (i.e., the user as whom that connection is bound). The target authorization identity for this control is specified as an "authzid" value, which should be either "dn:" followed by the distinguished name of the target user, or "u:" followed by the username.

Note that because of the inherent security risks associated with the use of the proxied authorization control, most directory servers that support its use enforce strict restrictions on the users that are allowed to request this control. If a user attempts to use the proxied authorization V2 request control and does not have sufficient permission to do so, then the server will return a failure response with the `AUTHORIZATION_DENIED` result code.

Configuring Proxied Authorization

Configuring proxied authorization requires a combination of access control instructions and the `proxied-auth` privilege to the entry that will perform operations as another user.



Note: You cannot use the `cn=Directory Manager` root DN as a proxying DN.

To Configure Proxied Authorization

1. Open a text editor and create a user entry, such as `uid=clientApp`, which is the user entry that will request operations as another user, `uid=admin,dc=example,dc=com`. The client application entry also requires the `proxied-auth` privilege to allow it to run proxied authorization requests. Save the file as `add-user.ldif`.

```
dn: ou=Applications,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: extensibleObject
ou: Admins
ou: Applications

dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Client
uid: clientApp
cn: Client App
sn: App
userPassword: password
ds-privilege-name: proxied-auth
```

2. Add the file using `ldapmodify`.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, some client may need access to an accounts subtree to retrieve customer information. Another client may need access to another subtree, such as a subscriber subtree. In this example, we want the client application to target the `ou=People,dc=example,dc=com` subtree. To allow the target, open a text editor and create an LDIF file to assign an ACI to that branch so that the client app user can access it as a proxy auth user. Note that the ACI should be on a single line of text. The example shows the ACI over multiple lines for readability. Add the file using the `ldapmodify`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
    userdn="ldap:///uid=clientApp,ou=Applications,dc=example,dc=com";)
```

- Run a search to test the configuration using the bind DN `uid=clientApp` and the `proxyAs` option, which requires that you prefix "dn:" to the proxying entry or "u:" to the username. The `uid=clientApp` binds to the server and proxies as `uid=admin` to access the `ou=People,dc=example,dc=com` subtree.

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"
```

Restricting Proxy Users

The Directory Server provides a set of operational attributes that restricts the proxied authorization capabilities of a client application and its proxyable target entry. When present in an entry, the Directory Server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes is present, then the user may potentially proxy as anyone.

The Directory Server supports a two-tier provision system that, when configured, can restrict specific users for proxied authorization. The first tier is a set of `ds-auth-may-proxy-as-*` operational attributes on the client entry that will bind to the server and carry out operations under the identity of another user. The second tier is a set of `ds-auth-is-proxyable-*` operational attributes on the user entry that defines whether access is allowed, prohibited, or required by means of proxied authorization. If allowed or required, the attributes define which client entries can proxy as the user.

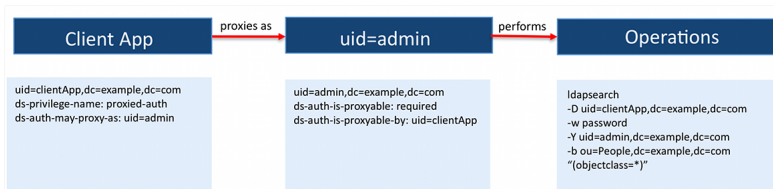


Figure 7: Proxying Operational Attributes

For example, if a client application, such as `uid=clientApp`, is requesting to search the `ou=People,dc=example,dc=com` branch as the user `uid=admin`, the command would look like this:

```
ldapsearch --bindDN uid=clientApp,dc=example,dc=com \
--bindPassword password \
--proxyAs uid=admin,dc=example,dc=com \
--baseDN ou=People,dc=example,dc=com \
"(object-class=*)"
```

At bind, the Directory Server evaluates the list of users in the `uid=clientApp` entry based on the presence of any `ds-auth-may-proxy-as-*` attributes. In the figure below, the `uid=clientApp` entry has a `ds-auth-may-proxy-as` attribute with a value, `uid=admin`, which means that the client app user may proxy only as the `uid=admin` account. Next, the server confirms that `uid=admin` is in the list of proxyable users and then evaluates the `ds-auth-is-proxyable-*` attributes present in the `uid=admin` entry. These attributes determine the list of restricted users that either are allowed, prohibited, or required to proxy as the `uid=admin` entry. In this case, the `uid=admin` entry has the `ds-auth-is-proxyable` attribute with a value of "required", which

indicates that the entry can only be accessed by means of proxied authorization. The `uid=admin` entry also has the `ds-auth-is-proxyable-by` attribute with a value of `uid=clientApp`, which indicates it can only be requested by the `uid=clientApp` entry. Once both sets of attributes have been confirmed, the `uid=clientApp` can bind to the server as the authenticated user. From this point, the Directory Server performs ACI evaluation on the branch to determine if the requested user has access rights to the branch. If the branch is accessible by the `uid=clientApp` entry, and then the search request is processed.

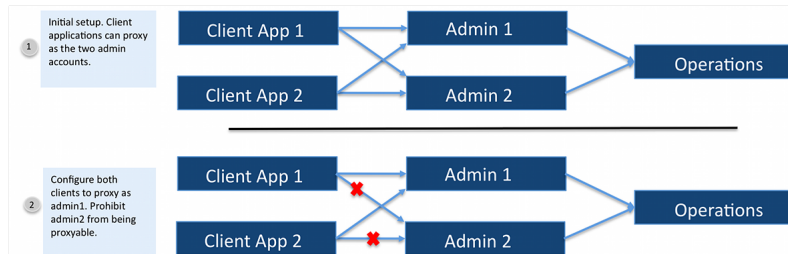


Figure 8: Proxying Operational Attributes Examples

About the `ds-auth-may-proxy-as-*` Operational Attributes

The Directory Server first evaluates the list of potential users that can be proxied for the authenticated user based on the presence of the `ds-auth-may-*` operational attributes in the entry. These operational attributes are multi-valued and are evaluated together if all are present in an entry:

- **`ds-auth-may-proxy-as`.** Specifies the user DNs that the associated user is allowed to proxy as. For instance, based on the previous example, you could specify in the `uid=clientApp` entry that it can proxy operations as `uid=admin` and `uid=agent1`.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin,dc=example,dc=com
ds-auth-may-proxy-as: uid=agent1,ou=admins,dc=example,dc=com
```

- **`ds-auth-may-proxy-as-group`.** Specifies the group DNs and its group members that the associated user is allowed to proxy as. For instance, you could specify that the potential users that the `uid=clientApp` entry can proxy as are those members who are present in the group `cn=Agents,ou=Groups,dc=example,dc=com`. This attribute is multi-valued, so that more than one group can be specified. Nested static and dynamic groups are also supported.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-group: cn=Agents,ou=Groups,dc=example,dc=com
```

- **`ds-auth-may-proxy-as-url`.** Specifies the DNs that are returned based on the criteria defined in an LDAP URL that the associated user is allowed to proxy as. For instance, the attribute specifies that the client can proxy as those entries that match the criteria in the LDAP URL. This attribute is multi-valued, so that more than one LDAP URL can be specified.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
```

```
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-url: ldap:///ou=People,dc=example,dc=com??sub?(l=austin)
```

About the ds-auth-is-proxyable-* Operational Attributes

After the Directory Server has evaluated the list of users that the authenticated user can proxy as, the server checks to see if the requested authorized user is in the list. If the requested authorized user is present in the list, then the server continues processing the proxiable attributes in the entry. If the requested authorized user is not present in the list, the bind will fail.

The operational attributes on the proxying entry are as follows:

- **ds-auth-is-proxyable.** Specifies whether the entry is proxyable or not. Possible values are: "allowed" (operation may be proxied as this user), "prohibited" (operations may not be proxied as this user), "required" (indicates that the account will not be allowed to authenticate directly but may only be accessed by some form of proxied authorization).
- **ds-auth-is-proxyable-as.** Specifies any users allowed to use this entry as a target of proxied authorization.
- **ds-auth-is-proxyable-as-group.** Specifies any groups allowed to use this entry as a target of proxied authorization. Nested static and dynamic groups are also supported.
- **ds-auth-is-proxyable-as-url.** Specifies the LDAP URLs that are used to determine any users that are allowed to use this entry as a target of proxied authorization.

Restricting Proxied Authorization for Specific Users

To illustrate how the proxied authorization operational attributes work, it is best to set up a simple example where two LDAP clients, `uid=clientApp1` and `uid=clientApp2` can freely proxy two administrator accounts, `uid=admin1` and `uid=admin2`. We will add the `ds-auth-may-proxy-as-*` and the `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization. For example, the two client applications will continue to proxy the `uid=admin1` account but the `uid=admin2` account will no longer be able to be used as a proxied entry.

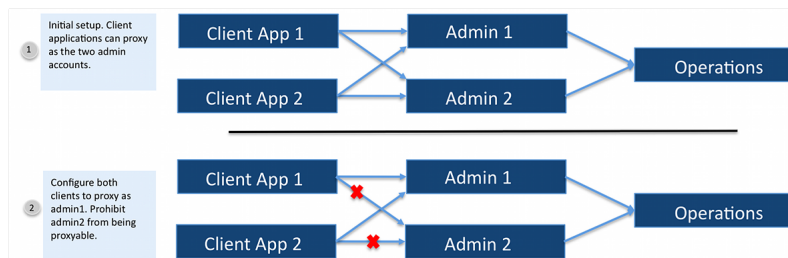


Figure 9: Proxy Users Example Scenario

To Restrict Proxied Authorization for Specific Users

1. For this example, set up two user entries, `uid=clientApp1` and `uid=clientApp2`, which will be proxying the `uid=admin1` and `uid=admin2` accounts to access the

ou=People,dc=example,dc=com subtree. Both entries have the proxied-auth privilege assigned to it. Open a text editor and create an LDIF file. Add the file using the ldapmodify tool. Note that "..." indicates that other attributes present in the entry are not included in the example for readability purposes.

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth

dn: uid=clientApp2,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
```

- Next, assign the ACI for each client application to the subtree, ou=People,dc=example,dc=com. Note that the ACIs should be on one line of text. The example displays the ACIs over multiple lines for readability.

```
dn: ou=People,dc=example,dc=com
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
      userdn="ldap:///uid=clientApp1,ou=Applications,dc=example,dc=com");
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
      userdn="ldap:///uid=clientApp2,ou=Applications,dc=example,dc=com");
```

- Run a search for each entry. In this example, assume that there are two admin accounts: admin1 and admin2 that have full access rights to user attributes. You should be able to proxy as the uid=admin1 and uid=admin2 entries to access the subtree for both clients.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin1,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"

$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

- Next, limit the proxied authorization capabilities for each client application. Update the uid=clientApp1 entry to add the ds-auth-may-proxy-as attribute. In this example, the ds-auth-may-proxy-as attribute specifies that uid=clientApp1 can proxy as the uid=admin1 entry. Open a text editor, create the following LDIF file, save it, and add it using ldapmodify. Note that ds-auth-may-proxy-as is multi-valued:

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as
ds-auth-may-proxy-as: uid=admin1,dc=example,dc=com
```

- Repeat the previous step for the uid=clientApp2 entry, except specify the ds-auth-may-proxy-as-url. The client entry may proxy as any DN that matches the LDAP URL.

```
dn: uid=clientApp2,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as-url
ds-auth-may-proxy-as-url: ldap:///dc=example,dc=com??sub?(uid=admin*)
```

- Next, we want to create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its `uniquemembers` to illustrate the use of the `ds-auth-proxyable-by-group` attribute. In this example, set up a static group using the `groupOfUniqueNames` object class.

```
dn: ou=Groups,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: groups

dn: cn=Client Applications,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Client Applications
ou: groups
uniquemember: uid=clientApp1,ou=Applications,dc=example,dc=com
uniquemember: uid=clientApp2,ou=Applications,dc=example,dc=com
```

- Update the `uid=admin1` entry to provide the DN that it may be proxied as. Add the `ds-auth-is-proxyable` and the `ds-auth-is-proxyable-by` attributes. For instance, we make the `uid=admin1` a required proxyable entry, which means that it can only be accessed by some form of proxied authorization. Then, specify each DN that can proxy as `uid=admin1` using the `ds-auth-is-proxyable-by`. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that the example includes all three types of `ds-auth-is-proxyable-by-*` attributes as an illustration, but, in an actual deployment, only one type of attribute is necessary if they all target the same entries.

```
dn: uid=admin1,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: required
-
add: ds-auth-is-proxyable-by
ds-auth-is-proxyable-by: ou=clientApp1,ou=Applications,dc=example,dc=com
ds-auth-is-proxyable-by: ou=clientApp2,ou=Applications,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-group
ds-auth-is-proxyable-by-group: cn=Client Applications,ou=Groups,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-url
ds-auth-is-proxyable-by-url: ldap:///ou=Applications,dc=example,dc=com??sub?
(uid=clientApp*)
```

- Next, prohibit proxying for the `uid=admin2` entry by setting the `ds-auth-is-proxyable` to prohibited. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`.

```
dn: uid=admin2,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: prohibited
```

- Run a search using the proxied account. For example, run a search first with `uid=clientApp1` or `uid=clientApp2` that proxies as `uid=admin1` to return a successful operation. However, if you run a search for `uid=clientApp1` that proxies as `uid=admin2`, as seen below, you will see an "authorization denied" message due to `uid=admin2` not matching the list of potential entries that can be proxied. The `ds-auth-may-proxy-as-*` attributes specify that the client can only proxy as `uid=admin1`:

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin2,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
```



```
"(objectclass=*)"
```

One of the operational attributes (ds-auth-may-proxy-as, ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry 'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that user to be proxied as user 'uid=admin2,dc=example,dc=com'

Result Code: 123 (Authorization Denied)

Diagnostic Message: One of the operational attributes (ds-auth-may-proxy-as, ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry 'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that user to be proxied as user 'uid=admin2,dc=example,dc=com'

10. Run another search using uid=clientApp2, which attempts to proxy as uid=admin2.

You will see an "authorization denied" message due to the presence of the ds-auth-is-proxyable:prohibited operational attribute, which states that uid=admin2 is not available for proxied authorization.

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin2,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"
```

The 'ds-auth-is-proxyable' operational attribute in user entry 'uid=admin2,dc=example,dc=com' indicates that user may not be accessed via proxied authorization

Result Code: 123 (Authorization Denied)

Diagnostic Message: The 'ds-auth-is-proxyable' operational attribute in user entry 'uid=admin2,dc=example,dc=com' indicates that user may not be accessed via proxied authorization

Chapter

17

Managing the Schema

This chapter presents a basic summary of the supported schema components on the Ping Identity Directory Server and procedures to extend the schema with new element definitions. The chapter presents the following topics:

Topics:

- [About the Schema](#)
- [About the Schema Editor](#)
- [Default Directory Server Schema Files](#)
- [Extending the Directory Server Schema](#)
- [General Tips on Extending the Schema](#)
- [Managing Attribute Types](#)
- [Creating a New Attribute over LDAP](#)
- [Managing Object Classes](#)
- [Managing an Object Class over LDAP](#)
- [Creating a New Object Class Using the Schema Editor](#)
- [Extending the Schema Using a Custom Schema File](#)
- [Managing Matching Rules](#)
- [Managing Attribute Syntaxes](#)
- [Using the Schema Editor Utilities](#)
- [Modifying a Schema Definition](#)
- [Deleting a Schema Definition](#)
- [Schema Checking](#)
- [Managing Matching Rule Uses](#)
- [Managing DIT Content Rules](#)
- [Managing Name Forms](#)
- [Managing DIT Structure Rules](#)
- [Managing JSON Attribute Values](#)
- [Configuring JSON Attribute Constraints](#)

About the Schema

A schema is the set of directory server rules that define the structures, contents, and constraints of a Directory Information Tree (DIT). The schema guarantees that any new data entries or modifications meet and conform to these predetermined set of definitions. It also reduces redundant data definitions and provides a uniform method for clients or applications to access its Directory Server objects.

The Ping Identity Directory Server ships with a default set of read-only schema files that define the core properties for the Directory Server. The Administrative Console provides a Schema Editor that administrators can use to view existing schema definitions and add new custom schema elements to their DIT. Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file will result in an "Unwilling to Perform" result.

About the Schema Editor

The Administrative Console Management Console provides a user-friendly graphical editor with tabs to manage any existing schema component related to the DIT: object classes, attributes, matching rules, attribute syntaxes, and schema utilities. The **Object Classes** and **Attribute Types** tabs enable viewing existing definitions as well as adding, modifying, or removing custom schema elements.

The **Matching Rules** and **Attribute Syntaxes** tabs are read-only and provide a comprehensive listing of all of the elements necessary to define new schema elements. The **Schema Utilities** tab provides a schema validator that allows you to load a schema file or perform a cut-and-paste operation on the schema definition to verify that it meets the proper schema and ASN.1 formatting rules. The **Utilities** tab also supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.

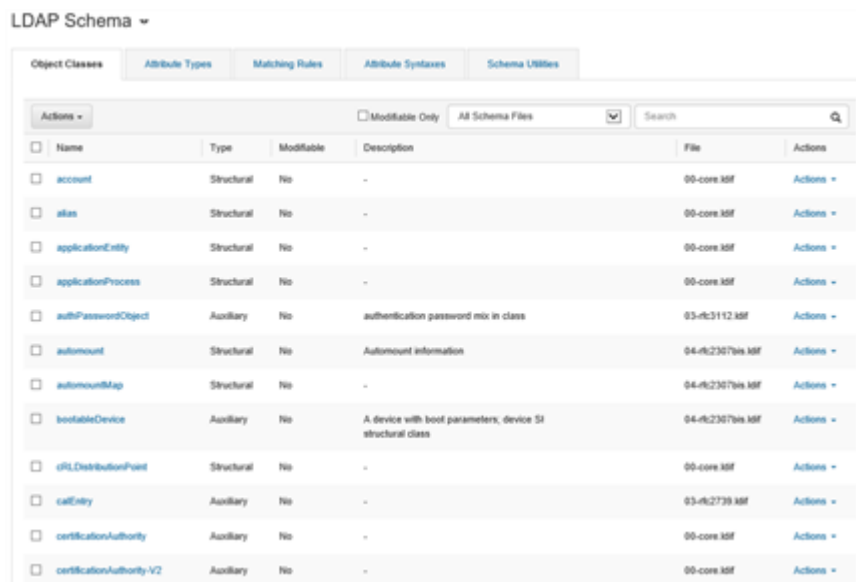


Figure 10: Example Schema Editor Screen

The Schema Editor provides two views for each definition: **Properties View** and **LDIF View**. The **Properties View** breaks down the schema definition by its properties and shows any inheritance relationships among the attributes. The **LDIF View** shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Default Directory Server Schema Files

The Ping Identity Directory Server stores its schema as a set of LDIF files for a Directory Server instance in the `<server-root>/config/schema` directory. The Directory Server reads the schema files in alphanumeric order at startup, so that the `00-core.ldif` file is read first, then `01-pwpolicy.ldif`, and then the rest of the files. Custom schema files should be named so that they are loaded in last. For example, custom schema elements could be saved in a file labelled `99-user.ldif`, which loads after the default schema files are read at startup.

The Directory Server then uses the schema definitions to determine any violations that may occur during add, modify, or import requests. Clients applications check the schema (i.e., matching rule definitions) to determine the assertion value algorithm used in comparison or search operations.

The default set of schema files are present at installation and should not be modified. Modifying the default schema files could result in an inoperable server.

The schema files have the following descriptions:

Table 32: Default Schema Files

Schema Files	Description
00-core.ldif	Governs the Directory Server's core functions.
01-pwpolicy.ldif	Governs password policies.
02-config.ldif	Governs the Directory Server's configuration.
03-changelog.ldif	Governs the Directory Server's change log.
03-rfc2713.ldif	Governs Java objects.
03-rfc2714.ldif	Governs Common Object Request Broker Architecture (CORBA) object references.
03-rfc2739.ldif	Governs calendar attributes for vCard.
03-rfc2926.ldif	Governs Server Location Protocol (SLP) mappings to and from LDAP schemas.
03-rfc2985.ldif	Governs PKCS #9 public-key cryptography.
03-rfc3112.ldif	Governs LDAP authentication passwords.
03-rfc3712.ldif	Governs printer services.
03-uddiv3.ldif	Governs web services registries of SOA components.
04-rfc2307bis.ldif	Governs mapping entities from TCP/IP and UNIX into X.500 entries.

Extending the Directory Server Schema

The Ping Identity Directory Server stores its schema as LDIF files in the `<server-root>/config/schema` directory. At startup, the Directory Server reads the schema files once in

alphanumeric order starting with `00-core.ldif` and ending with any custom schema definition files, such as `99-user.ldif` if present.

You can extend the schema to include additional customizations necessary for your Directory Server data using one of the following methods:

- **Using the Schema Editor.** This method is the easiest and quickest way to set up a schema definition and have it validated for the correct ASN.1 formatting. The Editor lets you define your schema properties, load your custom file, or perform a cut-and-paste operation on a new schema element. If any errors exist in the file, the Schema Editor generates an error message if the schema definitions do not pass compliance.
- **Using a Custom Schema File.** You can create a custom schema file with your new definitions using a text editor, save it as `99-user.ldif`, and then import the file using the Schema Editor or the `ldapmodify` tool. You must name the custom LDIF file with a high two-digit number prefix, so that the Directory Server will read the file **AFTER** the core schema files are read at startup. For example, you can name the file, `99-myschema.ldif`, etc. See the next section, [General Tips on Extending the Schema](#) to see the requirements for naming each file.
- **Using the Command Line.** If you have a small number of additions, you can extend the schema over LDAP and from the command line using the `ldapmodify` tool. The Directory Server writes the new schema changes to a file `99-user.ldif` in the `<server-root>/config/schema` directory. However, this method can be cumbersome as schema definitions require strict adherence to text spacing and white space characters.

General Tips on Extending the Schema

You should consider the following points when extending the schema:

- Never modify the default schema files as doing so could damage the Directory Server's processing capabilities.
- Define all attributes first before they can be used in an object class. If you are using the Schema Editor to add new schema elements, then you can use the Quick Add Attributes option when defining new objectclasses.
- Define the parent object class first before creating object classes that inherit from the parent.
- The schema file naming syntax requires that custom schema files must begin with exactly two digits followed by a non-digit character, followed by a zero or more characters and ending with ".ldif". Note that the two digits do not need to be followed by a dash ("-"). Any files that do not meet this criteria will be ignored and either a NOTICE or SEVERE_WARNING message will be logged.

Any file in the `<server-root>/config/schema` directory with a name that starts with "." or with a name that ends with a tilde (~), ".swp", or ".tmp" will generate a NOTICE message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria will generate a SEVERE_WARNING message indicating that it will be ignored.

- Define custom attributes and object classes in one file. Typically, this file will be the `99-user.ldif`. You can specify a different file name to which the Directory Server writes using the `X-SCHEMA-FILE` element and the file name in the definition. For example:

```
add: attributeTypes attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '99-custom.ldif' )
```

- Pay special attention to the white space characters in the schema definitions, where WSP means zero or more space characters, and SP means one or more space characters. The LDIF specification states that LDIF parsers should ignore exactly one space at the beginning of each continuation line, since continuation lines must begin with a space character. Thus, if you define a new schema definition with each keyword on a separate continuation line, you should add two spaces before an element keyword to be safe. For example, the following attribute definition has two spaces before the keywords: NAME, SUP, and X-ORIGIN.

```
attributeTypes: ( 2.5.4.32 NAME 'owner' SUP distinguishedName X-ORIGIN 'RFC 4519' )
```

- In a replicated topology, any new schema additions will be replicated to other replication servers to their respective Schema backend. The additions will be written to the file specified by the `X-SCHEMA-FILE` extension or written to `99-user.ldif` if no file is specified.

Managing Attribute Types

An attribute type determines the important properties related to an attribute, such as specifying the matching and syntax rules used in value comparisons. An attribute description consists of an attribute type and a set of zero or more options. Options are short, case-insensitive text strings that differentiate between attribute descriptions. For example, the LDAPv3 specification defines only one type of option, the tagging option, which can be used to tag language options, such as `cn;lang-de;lang-sp` or binary data, such as `userCertificate;binary`. You can also extend the schema by adding your own attribute definitions.

Attributes have the following properties:

- Attributes can be user attributes that hold information for client applications, or operational attributes that are used for administrative or server-related purposes. You can specify the purpose of the attribute by the `USAGE` element.
- Attributes are multi-valued by default. Multi-valued means that attributes can contain more than one value within an entry. Include the `SINGLE-VALUE` element if the attribute should contain at most one value within an entry.
- Attributes can inherit properties from a parent attribute as long as they both have the same `USAGE`, and the child attribute has the same `SYNTAX` or its `SYNTAX` allows values which are a subset of the values allowed by the `SYNTAX` of the parent attribute. For example, the `sn` attribute is a child of the `name` attribute.

Attribute Type Definitions

New attribute types do not require server code extensions if the provided matching rules and attribute syntaxes are used in the definitions. Administrators can create new attributes using the Schema Editor, which stores the definition in a file in the <server-root>/config/schema directory. See *Extending the Directory Server Schema* for more information.

The formal specification for attribute types is provided in RFC 4512, section 4.1.2 as follows:

```
AttributeTypeDescription = "(" wsp; Left parentheses followed by a white space
numericoid                ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]   ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]  ; Optional descriptive string
[ sp "OBSOLETE" ]         ; Determines if the element is active
[ sp "SUP" sp oid ]       ; Specifies the supertype
[ sp "EQUALITY" sp oid ]   ; Specifies the equality matching rule
[ sp "ORDERING" sp oid ]   ; Specifies ordering matching rule
[ sp "SUBSTR" sp oid ]     ; Specifies substrings matching rule
[ sp "SYNTAX" sp oidlen ] ; Numeric attribute syntax with minimum upper bound
                           ; length expressed in {num}
[ sp "SINGLE-VALUE" ]      ; Specifies if the attribute is single valued in
                           ; the entry
[ sp "COLLECTIVE" ]       ; Specifies if it is a collective attribute
[ sp "NO-USER-MODIFICATION" ] ; Not modifiable by external clients
[ sp "USAGE" sp usage ]   ; Application usage
extensions wsp ")"        ; Extensions followed by a white space and ")"

usage = "userApplications" / ; Stores user data
       "directoryOperation" / ; Stores internal server data
       "distributedOperation" / ; Stores operational data that must be synchronized
                           ; across servers
       "dSAOperation"       ; Stores operational data specific to a server and
                           ; should not be synchronized across servers
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" / ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" / ; Specifies which schema file contains the definition
"X-APPROX" / ; Specifies the approximate matching rule
"X-ALLOWED-VALUE" / ; Explicitly specifies the set of allowed values
"X-VALUE-REGEX" / ; Specifies the set of regular expressions to compare against
                  ; attribute values to determine acceptance
"X-MIN-VALUE-LENGTH" / ; Specifies the minimum character length for attribute values
"X-MAX-VALUE-LENGTH" / ; Specifies the maximum character length for attribute values
"X-MIN-INT-VALUE" / ; Specifies the minimum integer value for the attribute
"X-MAX-INT-VALUE" / ; Specifies the maximum integer value for the attribute
"X-MIN-VALUE-COUNT" / ; Specifies the minimum number of allowable values for the
                  ; attribute
"X-MAX-VALUE-COUNT" / ; Specifies the maximum number of allowable values for the
                  ; attribute
"X-READ-ONLY" ; True or False. Specifies if the file that contains the
                  ; schema element is marked as read-only in the server
                  ; configuration.
```

Basic Properties of Attributes

The Basic Properties section displays the standard elements in schema definition.

Table 33: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.

Attributes	Description
Description	Specifies an optional definition that describes the attribute and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Syntax	Specifies the attribute syntax used. For example, the <code>userPassword</code> attribute uses the User Password Syntax whereas the <code>authPassword</code> attribute uses the Authentication Password Syntax.
Parent	Specifies the schema definition's parent or supertype if any. The analogous LDIF equivalent is "SUP".
Multivalued	Specifies if the attribute can appear more than once in its containing object class.
Required By Class	Specifies any object classes that require the attribute.
Allowed By Class	Specifies any object classes that can optionally use the attribute.
Value Restrictions	Specifies any restriction on the value of the attribute.

The Extra Properties section provides additional auxiliary information associated with the attribute.

Table 34: Basic Properties of Attributes

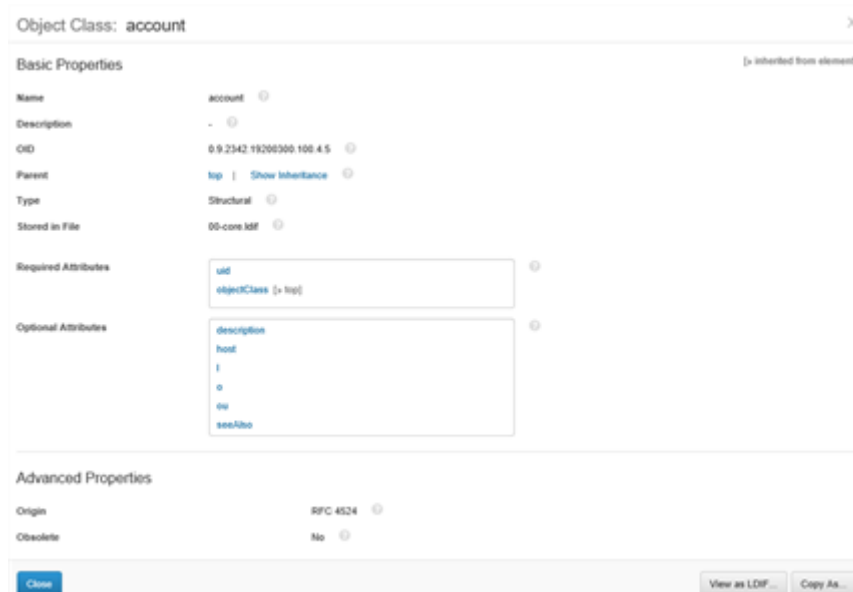
Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element. For example, NAME ('sn' 'surname').
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Stored in File	Specifies the schema file that stores the definition in the <code><server-root>/config/schema</code> folder.
Usage	Specifies the intended use of the attribute. Choices are the following: <ul style="list-style-type: none"> > userApplications > directoryOperation > distributedOperation > dSAOperation
User-Modifiable	Specifies if the attribute can be modified by an authorized user.
Obsolete	Specifies if the schema definition is obsolete or not.
Matching Rules	Specifies the associated matching rules for the attribute.

Viewing Attributes

The Schema Editor displays all of the attribute types on your directory server instance. It shows the basic properties that are required elements plus the extra properties that are allowed within the attribute definition.

To View Attribute Types Using the Schema Editor

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. In the Administrative Console **Schema Editor**, click the **Attribute Types** tab.
4. Click a specific attribute to view its definition. In this example, click the `account` attribute. In the **Object Class** window, view the attribute properties.



5. Click the **View as LDIF** button to see the equivalent attribute definition in ASN.1 format.

To View Attribute Types over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute `attributeTypes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

To View a Specific Attribute Type over LDAP

- Use `ldapsearch` with the `--dontWrap` option and use the `grep` command to search for a specific attribute.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a New Attribute over LDAP

The following section shows how you can add the schema element from the previous section over LDAP. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting.

To Add a New Attribute to the Schema over LDAP

1. Create an LDIF file with the new attribute definition using a text editor. Save the file as `myschema.ldif`.

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
    EQUALITY booleanMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
    SINGLE-VALUE
    USAGE userApplications
    X-ALLOWED-VALUES 'Y' 'N' 'y' 'n'
    X-ORIGIN 'Directory Server Example' )
```

2. Use `ldapmodify` to add the attribute.

```
$ bin/ldapmodify --filename myschema.ldif
```

3. Verify the addition by displaying the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
--dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. You can view the custom schema file at `<server-root>/config/schema/99-user.ldif`. You should see the following:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( contractorStatus-OID
    NAME 'contractorStatus'
    EQUALITY booleanMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
    SINGLE-VALUE
    USAGE userApplications
    X-ORIGIN 'PingData Directory Server Example' )
```

To Add Constraints to Attribute Types

- The Directory Server provides attribute type extensions that constrain the values for the associated attribute using the `DirectoryString` attribute syntax. The following schema definition includes two `attributeType` definitions for `myAttr1` and `myAttr2`. The first definition constrains the values for the attribute `myAttr1` to 'foo', 'bar', 'baz'. The second

definition constrains the minimum allowable length for `myAttr2` to 1 and the maximum allowable length to 5.

```
attributeTypes: (1.2.3.4
  NAME 'myAttr1'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ALLOWED-VALUES ( 'foo' 'bar' 'baz' ))
attributeTypes: ( 1.2.3.5
  NAME 'myAttr2'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-MIN-VALUE-LENGTH '1'
  X-MAX-VALUE-LENGTH '5' )
```

Managing Object Classes

Object classes are sets of related information objects that form entries in a Directory Information Tree (DIT). The Directory Server uses the schema to define these entries, to specify the position of the entries in a DIT, and to control the operation of the server. You can also extend the schema by adding your own schema definitions.

Object classes have the following general properties:

- Object classes must have a globally unique name or identifier.
- Object classes specify the required and allowed attributes in an entry.
- Object classes can inherit the properties and the set of allowed attributes from its parent object classes, which may also be part of a hierarchical chain derived from the top abstract object class.
- Object classes that are defined in the Ping Identity Directory Server can be searched using the `objectClasses` operational attribute. The Directory Server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object Classes Types

Based on RFC 4512, object classes can be a combination of three different types:

- **Abstract object classes** are used as the base object class, from which structural or auxiliary classes inherit its properties. This inheritance is a one-way relationship as abstract object classes cannot be derived from structural or auxiliary classes. The most common abstract object class is `top`, which defines the highest level object class in a hierarchical chain of object classes.
- **Structural object classes** define the basic attributes in an entry and define where an entry can be placed in a DIT. All entries in a DIT belong to one structural object class. Structural object classes can inherit properties from other structural object classes and from abstract object classes to form a chain of inherited classes. For example, the `inetOrgPerson` structural object class inherits properties from the `organizationalPerson` structural class, which inherits from another object class, `person`.

- **Auxiliary object classes** are used together with structural object classes to define additional sets of attributes required in an entry. The auxiliary object class cannot form an entry alone but must be present with a structural object class. Auxiliary object classes cannot derive from structural object classes or vice-versa. They can inherit properties from other auxiliary classes and from abstract classes.

Object Class Definition

New object classes can be specified with existing schema components and do not require additional server code extensions for their implementation. Administrators can create new object classes using the Schema Editor, which manages schema in the <server-root>/config/schema directory. See [Extending the Directory Server Schema](#) for more information.

The object class definition is defined in RFC 4512, section 4.1.1, as follows::

```
ObjectClassDescription = "(" wsp; Left parenthesis followed by a white space
numericoid              ; Required numeric object identifier
[ sp "NAME" sp qdescrs ] ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ] ; Optional descriptive string
[ sp "OBSOLETE" ]       ; Determines if the element is inactive
[ sp "SUP" sp oid ]     ; Specifies the direct superior object class
[ sp kind ]             ; abstract, structural (default), auxiliary
[ sp "MUST" sp oids ]   ; Required attribute types
[ sp "MAY" sp oids ]    ; Allowed attribute type
extensions wsp ")"     ; Extensions followed by a white space and ")"

usage = "userApplications" / ; Stores user data
      "directoryOperation" / ; Stores internal server data
      "distributedOperation" / ; Stores operational data that must be synchronized
                                ; across servers
      "dSAOperation"         ; Stores operational data specific to a server and
                                ; should not be synchronized across servers
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" / ; Specifies where the object class is defined
"X-SCHEMA-FILE" / ; Specifies which schema file contains the definition
"X-READ-ONLY" / ; True or False. Specifies if the file that contains
                 ; the schema element is marked as read-only
                 ; in the server configuration.
```



Note: Although RFC 4512 allows multiple superior object classes, the Ping Identity Directory Server allows at most one superior object class, which is defined by the SUP element in the definition.

Basic Object Class Properties

The Basic Properties section displays the standard elements in schema definition.

Table 35: Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the object class and its contents. The analogous LDIF equivalent is "DESC".

Attributes	Description
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Parent	Specifies the schema definition's hierarchical parent or superior object class if any. An object class can have one parent. The analogous LDIF equivalent
Type	Specifies the type of schema definition: abstract, structural, or auxiliary. The analogous LDIF equivalent is "ABSTRACT", "STRUCTURAL", or "AUX"
Required Attributes	Specifies any required attributes with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MUST".
Optional Attributes	Specifies any optional attributes that could be used with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MAY".

The Extra Properties section provides additional auxiliary information associated with the object class.

Table 36: Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element although most object classes do not have aliases.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Obsolete	Specifies if the schema definition is obsolete or not.
Stored in File	Specifies the schema file that stores the definition in the <server-root>/config/schema folder.

Viewing Object Classes

You can view the object classes on your Directory Server by using the Administrative Console Schema Editor, over LDAP using the `ldapsearch` tool, or some third party tool. The Schema Editor displays all of the object classes on the directory server instance. It shows the basic properties that are required elements and the extra properties that are allowed within the object class.

To View Object Classes over LDAP

- Use `ldapsearch` tool to view a multi-valued operational attribute, `objectClasses`, which publishes the object class definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
--dontWrap "(objectclass=*)" objectClasses
```

```
dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST aliasedObjectName
X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
MAY ( searchGuide $ description ) X-ORIGIN 'RFC 4519' )
...(more output)...
```

Managing an Object Class over LDAP

The following section shows how you can manage an object class schema element over LDAP by adding a new attribute element to an existing object class. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting. The following example procedure adds an attribute, `contractorAddress`, to the custom schema file, then adds it to the `contractor` objectclass.

To Manage an Object Class over LDAP

1. Assume that you have defined the `contractorAddress` attribute, create an LDIF file called `contractorAddress-attr.ldif` with the following content:

```
dn: cn=schema
changetype: modify

add: attributeTypes attributeTypes: ( contractor-OID NAME 'contractorAddress'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
USAGE userApplications
X-ORIGIN 'user defined'
X-SCHEMA-FILE '98-custom-schema.ldif' )
X-ORIGINS 'user defined'
X-SCHEMA-FILE '98-custom-schema.ldif' )
```

2. Add the attribute using `ldapmodify`.

```
$ bin/ldapmodify --filename contractorAddress-attr.ldif
```

3. Next, create an LDIF file to modify the `contractor` objectclass to allow this attribute. When doing this, you are just re-adding the updated `objectClass` and the Directory Server will handle the proper replacement of the existing object class with the new one. Create a file called `contractor-oc.ldif`. Make sure that the lines are not wrapped, the `objectClasses` line should be one continuous line.

```
dn:cn=schema
changetype: modify
add: objectClasses
objectClasses: ( contractor-OID NAME 'contractor'
DESC 'Contractor status information
SUP top
AUXILIARY MAY ( contractorStatus $ contractorAgency $ contractorAddress )
X-ORIGIN 'Directory Server Example'
X-SCHEMA-FILE '98-custom-schema.ldif' )
```

4. Update the `objectClass` using `ldapmodify` as follows:

```
$ bin/ldapmodify --filename contractor-oc.ldif
```

5. These schema changes will be replicated to all servers in the replication topology. Verify the change by looking at the `config/schema/98-custom-schema.ldif` file on the other servers in the replication topology to ensure that the changes are present.
6. If you need to add an index for this attribute, you can do so by using the `dsconfig` command-line utility. You will need to do this on each server in your topology unless you have server configuration groups set up. See [Configuring Server Groups](#) for more information.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \  
--index-name contractorAddress --set index-type:equality
```

7. Rebuild the index online. This will not affect other indexes or entries since there is no currently existing data for this attribute on any entry.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress
```

Creating a New Object Class Using the Schema Editor

The procedures to create a new object class are similar to that of creating a new attribute. Make sure that any attributes that are part of the new object class are defined prior to defining the object class.

To Create a New Object Class Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab, and then click **New** located in the bottom left of the window.
4. Enter the properties for the new objectclass. In the **Attributes** box, filter the types of attributes required for the new object class. Click the right arrow to move it into the **Required** or the **Optional** box. All custom attributes appear at the bottom of the list in the **Attributes** box.

The screenshot shows the 'New Object Class' dialog box. It has a title bar with 'New Object Class' and a close button. Below the title bar is a checkbox for 'Show Advanced Properties'. The main area is divided into sections: 'Basic Properties' and 'Attributes'. Under 'Basic Properties', there are fields for 'OID' (111), 'Name' (ExampleClass), 'Description' (optional), 'Parent' (domain), 'Type' (Structural), and 'Stored in File' (99-user.ldf). Under 'Attributes', there is a search bar and a list of attributes. The list is divided into 'Required Attributes' (dc, objectClass) and 'Optional Attributes' (associatedName). There are also 'Save' and 'Cancel' buttons at the bottom.

Extending the Schema Using a Custom Schema File

You can add new attributes and object classes to your Directory Server schema by creating a custom schema file. You can import the file using the Schema Editor, over LDAP using the `ldapmodify` tool, or from the command line. Make sure to define the attributes first, then define the object classes.

To Extend the Schema Using a Custom Schema File

1. Create an LDIF file with the new attribute extensions using a text editor.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example' )
attributeTypes: ( contractorAgency-OID NAME 'contractorAgency'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Ping Identity Directory Server Example' )
```

2. In the same LDIF file, add a new object class definition after the attribute types. In this example, create an auxiliary object class, `contractor`, that alone cannot be used as an entry. The object class will be used to add supplemental information to the `inetOrgPerson` structural object class. The attributes are all optional for the new object class.

```
objectClasses: ( contractor-OID
```

```
NAME 'contractor'
DESC 'Contractor status information'
SUP top
AUXILIARY
MAY ( contractorStatus $ contractorAgency )
X-ORIGIN 'Ping Identity Directory Server Example' )
```

3. Save the file as `99-auxobjclass.ldif`. and place it in the `<server-root>/config/schema` directory.
4. At this stage, the schema extensions are not loaded into the Directory Server yet. You have four options to load them:
 - Create a task that loads the new extensions into the schema. We create a task labelled with the ID "add-schema-99-auxobjclass" and add it using `ldapmodify`. The server does not need to be restarted using this method.

```
dn: ds-task-id=add-schema-99-auxobjclass,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-add-schema-file
ds-task-id: add-schema-99-auxobjclass
ds-task-class-name: com.unboundid.directory.server.tasks.AddSchemaFileTask
ds-task-schema-file-name: 99-auxobjclass.ldif
```

- Import the schema file using the Administrative Console Schema Editor. You do not need to restart the server when using this method.
- Place the `99-auxobjclass.ldif` file in the `<server-root>/config/schema` directory and restart the Directory Server. The schema file is read at startup.
- Add the schema file using `load-ldap-schema-file`. You do not need to restart the server when using this method.

```
$ bin/load-ldap-schema-file --schemaFile config/schema 99-auxobjclass.ldif
```

5. Verify the addition by displaying the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.9)" contractorStatus
```

```
dn: uid=user.9,ou=People,dc=example,dc=com
contractorStatus: TRUE
```

Managing Matching Rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations. They are also used in evaluating search filter elements including distinguished names and attributes. Matching rules are defined for each attribute based on EQUALITY (e.g., two attributes are equal based on case, exact match, etc.), SUBSTR (e.g., assertion value is a substring of an attribute), and ORDERING (e.g., greater than or equal, less than or equal, etc.) properties.



Note: The Ping Identity Directory Server supports an APPROXIMATE matching rule that compares similar attributes based on fuzzy logic. Thus, attributes that are similar or "sound-like" each other are matched. For example, "petersen" would match "peterson".

Matching Rule Definition

New matching rules require additional server code extensions to be implemented on the Ping Identity Directory Server. If you need new matching rules, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.3 as follows:

```
MatchingRuleDescription = "(" wsp           ; Left parentheses followed by a white space
numericoid                ; Required numeric object identifier identifying
                           ;   this matching rule
[ sp "NAME" sp qdescrs    ; Short name descriptor
[ sp "DESC" sp qdstring   ; Description
[ sp "OBSOLETE" ]        ; Specifies if the rule is inactive
sp "SYNTAX" sp numericoid ; Assertion syntax
extensions wsp ")"       ; Extensions followed by a white space and ")"
```

Default Matching Rules

The Ping Identity Directory Server provides a large set of matching rules, which support a variety of applications. The default matching rules available for the Directory Server are listed in the table below for each matching rule type: Equality, Substring, Ordering, and Approximate matches.

Table 37: Default Matching Rules

Matching Rule/OID	Attribute Syntax/OID	Description
uuidMatch/ 1.3.6.1.1.16.2	UUID/ 1.3.6.1.1.16.1	Compares an asserted UUID with a stored UUID attribute value for equality. RFC 4530.
uuidOrderingMatch/ 1.3.6.1.1.16.3	UUID/ 1.3.6.1.1.16.1	Compares the collation order of an asserted UUID with a stored UUID attribute value for ordering RFC 4530.
caseExactIA5Match/ 1.3.6.1.4.1.1466.109.114.1	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. RFC 4517.
caseIgnoreIA5Match/ 1.3.6.1.4.1.1466.109.114.2	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreIA5SubstringsMatch/ 1.3.6.1.4.1.1466.109.114.3	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
authPasswordExactMatch/ 1.3.6.1.4.1.4203.1.2.2	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password exact matching rule.

Matching Rule/OID	Attribute Syntax/OID	Description
authPasswordMatch/ 1.3.6.1.4.1.4203.1.2.3	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password matching rule.
ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Syntax based on the phonetic Double Metaphone algorithm for approximate matching.
ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password exact matching rule.
ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password equality matching rule.
historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4	1.3.6.1.4.1.30221.1.3.5	Compares the collation order of a historical change sequence number with a historical CSN attribute value.
caseExactIA5SubstringsMatch/ 1.3.6.1.4.1.30221.1.4.902	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. RFC 4517.
compactTimestampMatch/ 1.3.6.1.4.1.30221.2.4.1	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compact Timestamp matching rule.
compactTimestampOrderingMatch/ 1.3.6.1.4.1.30221.2.4.2	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compares the collation order of a compact timestamp number with an attribute value of Compact Timestamp syntax.
objectIdentifierMatch/ 2.5.13.0	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Compares an asserted value with an attribute value of OID syntax. RFC 4517.
distinguishedNameMatch/ 2.5.13.1	DN/ 1.3.6.1.4.1.1466.115.121.1.12	Compares an asserted value with an attribute value of DN syntax. Spaces around commas and semicolons are ignored. Spaces around plus and equal signs around RDN components are ignored. RFC 4517.
caseIgnoreMatch/ 2.5.13.2	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreOrderingMatch/ 2.5.13.3	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseIgnoreSubstringsMatch/ 2.5.13.4	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring value with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are

Matching Rule/OID	Attribute Syntax/OID	Description
		treated as a single space. RFC 4517
caseExactMatch/ 2.5.13.5	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value of Directory String syntax. RFC 4517.
caseExactOrderingMatch/ 2.5.13.6	Directory String 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. RFC 3698
caseExactSubstringsMatch/ 2.5.13.7	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Directory String syntax. RFC 3698
numericStringMatch/ 2.5.13.8	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringOrderingMatch/ 2.5.13.9	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares the collation order of the asserted string with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringSubstringsMatch/ 2.5.13.10	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
caseIgnoreListMatch/ 2.5.13.11	Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41	Compares an asserted value with an attribute value which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreListSubstringsMatch/ 2.5.13.12	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698.
booleanMatch/ 2.5.13.13	Boolean/ 1.3.6.1.4.1.1466.115.121.1.7	Compares an asserted boolean value with an attribute value of BOOLEAN syntax. Returns true if the values are both TRUE or both FALSE. RFC 3698.
integerMatch/ 2.5.13.14	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares an asserted value with an attribute value of INTEGER syntax. RFC 4517.

Matching Rule/OID	Attribute Syntax/OID	Description
integerOrderingMatch/ 2.5.13.15	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares the collation order of the asserted integer with an attribute value of Integer syntax. Returns true if the attribute value is less than the asserted value. RFC 3698.
bitStringMatch/ 2.5.13.16	Bit String/ 1.3.6.1.4.1.1466.115.121.1.6	Compares an asserted Bit String value with an attribute value of Bit String syntax. RFC 4517.
octetStringMatch/ 2.5.13.17	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares an asserted value with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
octetStringOrderingMatch/ 2.5.13.18	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares the collation order of the asserted octet string with an attribute value of Octet String syntax. Zero precedes a one bit. Shorter strings precede longer strings. RFC 3698.
octetStringSubstringsMatch/ 2.5.13.19	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
telephoneNumberMatch/ 2.5.13.20	Telephone Number/ 1.3.6.1.4.1.1466.115.121.1.50	Compares an asserted value with an attribute value of Telephone Number syntax. RFC 4517.
telephoneNumberSubstringsMatch/ 2.5.13.21	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted value with the substrings of an attribute value of Telephone Number String syntax. RFC 4517.
presentationAddressMatch/ 2.5.13.22	Presentation Address/ 1.3.6.1.4.1.1466.115.121.1.43	Compares an asserted value with an attribute value of Presentation Address syntax. RFC 4517.
uniqueMemberMatch/ 2.5.13.23	Name and Optional UID/ 1.3.6.1.4.1.1466.115.121.1.34	Compares an asserted value with an attribute value of Unique Member syntax. RFC 4517.
protocolInformationMatch/ 2.5.13.24	Protocol Information/ 1.3.6.1.4.1.1466.115.121.1.42	Compares an asserted value with an attribute value of Protocol Information syntax. RFC 4517.
generalizedTimeMatch/ 2.5.13.27	Generalized Time/ 1.3.6.1.4.1.1466.115.121.1.24	Compares an asserted value with an attribute value of Generalized Time syntax. RFC 4517.
generalizedTimeOrderingMatch/ 2.5.13.28	Generalized Time 1.3.6.1.4.1.1466.115.121.1.24	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored. RFC 4517.
integerFirstComponentMatch/ 2.5.13.29	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Equality matching rules for subschema attributes between

Matching Rule/OID	Attribute Syntax/OID	Description
		an Integer syntax and the value syntax. RFC 4517.
objectIdentifierFirstComponentMatch/ 2.5.13.30	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Equality matching rules for subschema attributes between an OID syntax and the value syntax. RFC 4517.
directoryStringFirstComponentMatch/ 2.5.13.31	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted Directory String value with an attribute value of type SEQUENCE whose first component is mandatory and of type Directory String. Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of caseIgnoreMatch. RFC 3698.
wordMatch/ 2.5.13.32	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted word with any word in the attribute value for equality RFC 3698.
keywordMatch/ 2.5.13.33	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with any keyword in the attribute value for equality. RFC 3698.

Basic Matching Rule Properties

The Properties section displays the standard elements in a matching rule schema definition.

Table 38: Basic Properties of Matching Rules

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Specifies an optional definition that describes the matching rule. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Type	Specifies the type of type of matching rule: Equality, Ordering, Substring, or Approximate.
Syntax	Specifies the matching rule syntax.
Used by Attributes	Specifies any attributes that use the corresponding matching rule.

Viewing Matching Rules

You can view the matching rules on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the matching rules on your Directory Server instance. It shows the basic properties that are allowed within the matching rule.

To View Matching Rules Over LDAP

- Use `ldapsearch` to view a multi-valued operational attribute, `matchingRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRules
```

Managing Attribute Syntaxes

The attribute type definition has a `SYNTAX` element, or attribute syntax, that specifies how the data values for the attribute are represented. The syntax can be used to define a large range of data types necessary for client applications. An attribute syntax uses the Abstract Syntax Notation One (ASN.1) format for its definitions.

Attribute Syntax Definition

New attribute syntaxes require additional code to be implemented on the Ping Identity Directory Server. If you need new syntax definitions, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.5 as follows:

```
SyntaxDescription = "(" wsp
numericoid        ; Object identifier
[ sp "DESC" sp qdstring ] ; Description
extensions wsp ")" ; Extensions followed by a white space and ")"
```

Default Attribute Syntaxes

The Ping Identity Directory Server supports a large set of Attribute Syntax rules for applications. The default Attribute Syntax rules available for the directory server are listed in the table below.

Table 39: Default Attribute Syntaxes

LDAP Syntax	OID	Description
UUID	1.3.6.1.1.16.1	128-bit (16 octets) Universally Unique Identifier (UUID) used for Uniform Resource Names as defined in RFC 4122. For example, a4028c1a-f36e-11da-ba1a-04112154bd1e.
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3	Syntax for the AttributeTypeDescription rule based on RFC 4517.
Binary	1.3.6.1.4.1.1466.115.121.1.5	Strings based on Basic Encoding Rules (BER) or Distinguished Encoding rules (DER). For example, an X.509 digital certificate or LDAP messages are BER encoded.

LDAP Syntax	OID	Description
Bit String	1.3.6.1.4.1.1466.115.121.1.6	Sequence of binary digits based on RFC 4517. For example, '0010111'B.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	TRUE or FALSE.
Certificate	1.3.6.1.4.1.1466.115.121.1.8	BER/DER-encoded octet strings based on an X.509 public key certificate as defined in RFC 4523.
Certificate List	1.3.6.1.4.1.1466.115.121.1.9	BER/DER-encoded octet string based on an X.509 certificate revocation list as defined in RFC 4523.
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10	BER/DER-encoded octet string based on an X.509 public key certificate pair as defined in RFC 4523.
Country String	1.3.6.1.4.1.1466.115.121.1.11	Two character country code specified in ISO 3166. For example, US, CA, etc.
DN	1.3.6.1.4.1.1466.115.121.1.12	Distinguished name of an entry as defined in RFC4514.
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	Sequence of services in preference order by which an entity receives messages as defined in RFC4517. For example, videotext \$ telephone.
Directory String	1.3.6.1.4.1.1466.115.121.1.15	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string.
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16	DITContentRuleDescription as defined in RFC4517.
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17	DITStructureRuleDescription as defined in RFC4517.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	Combination of attribute types and filter operators to be used to construct search filters as defined in RFC4517. For example, person#(sn \$EQ)#oneLevel.
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22	Fax telephone number on the public switched telephone network as defined in RFC4517.
Fax	1.3.6.1.4.1.1466.115.121.1.23	Image generated using Group 3 fax process as defined in RFC4517.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	String representing data and time as defined in RFC4517. YYYYMMDDHHMMSS[.,fraction] [(+ -HHMM)]Z For example, 201103061032, 201103061032-0500, or 201103061032Z ("Z" indicates Coordinated Universal Time).
Guide	1.3.6.1.4.1.1466.115.121.1.25	Attribute types and filter operators as defined in RFC4517.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	String of zero or more characters from the International Alphabet 5 (IA5) character set as defined in RFC4517.
Integer	1.3.6.1.4.1.1466.115.121.1.27	String representations of integer values. For example, the character string "1234" represents the number 1234 as defined in RFC4517.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	Image in JPEG File Interchange Format (JFIF) as defined in RFC4517.
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30	MatchingRuleDescription as defined in RFC4512.

LDAP Syntax	OID	Description
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31	Attribute types to which a matching rule is applied in an extensibleMatch search filter (RFC4511).
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	Distinguished name and an optional unique identifier that differentiates identical DNs as defined in RFC4517. For example, uid=jsmith,ou=People,dc=example,dc=com#'0111'B
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35	NameFormDescription as defined in RFC4512.
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Sequence of one or more numerals and spaces as defined in RFC4517. For example, 14 848 929 102.
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37	ObjectClassDescription as defined in RFC 4512.
OID	1.3.6.1.4.1.1466.115.121.1.38	Object identifier as defined in RFC 4512.
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39	Specifies an electronic mailbox as defined in RFC 4517. For example, otherMailbox = google \$ user@gmail.com
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Sequence of zero or more octets (8-bit bytes) as defined in RFC4517.
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	Strings of characters that form a multi-line address in a physical mail system. Each component is separated by a "\$". For example, 1234 Main St. \$Austin, TX 78744\$USA.
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42	Undefined.
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43	String encoded OSI presentation address as defined in RFC 1278. For example, TELEX +00728722+RFC-1006+03+10.0.0.6
Printable String	1.3.6.1.4.1.1466.115.121.1.44	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517.
RFC3672 Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45	Syntax based on subtree specification as defined as RFC 3672.
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a SupportedAlgorithm ASN.1 value.
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50	String of printable international telephone number representations in E.123 format as defined in RFC 4517. For example, +1 512 904 5525.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	Identifier and telex terminal as defined in RFC 4517.
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	String representing the telex number, country code, and answerback code as defined in RFC 4517. For example, 812374, ch, ehg.
UTC Time	1.3.6.1.4.1.1466.115.121.1.53	Character string representing the data and time in UTC Time format as defined as RFC 4517: YYMMDDHHMM[SS][(+ -)HHMM]Z], where Z is the coordinated universal time. For example, 0903051035Z, 0903051035-0500.

LDAP Syntax	OID	Description
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54	SyntaxDescription as defined in RFC4512.
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58	Syntax for assertion values in an extensible match as defined in RFC 4517.
Authentication Password Syntax	1.3.6.1.4.1.4203.1.1.2	Encoded password storage syntax as defined in RFC 3112. For example, the syntax specifies the storage scheme in brackets as follows: <storage-scheme>\${auth component}\${auth value} For example: SSHA\$xdEZRqgyJk=\$egDEFDXvdeeEnXUEIDPnd39dkpe=
User Password Syntax	1.3.6.1.4.1.30221.1.3.1	Encoded password storage syntax as defined in RFC 2307. For example, the syntax specifies the storage scheme in brackets as follows: {SSHA}XaljOF0ii3fOwCrU1klgBpWFayqSYs+5W1pMnw==
Relative Subtree Specification	1.3.6.1.4.1.30221.1.3.2	Similar to the RFC 3672 subtree specification except it uses an LDAP search filter as the specification filter.
Absolute Subtree Specification	1.3.6.1.4.1.30221.1.3.3	Syntax for a subset of entries in a subtree based on RFC 3672.
Sun-defined Access Control Information	1.3.6.1.4.1.30221.1.3.4	Syntax for access control instructions used in Sun Directory Servers.
Compact Timestamp	1.3.6.1.4.1.30221.2.3.1	Syntax based on compact timestamp ISO 8601 format. For example, 20110306T102532.

Basic Attribute Syntax Properties

The Properties section displays the standard elements in an attribute syntax.

Table 40: Basic Properties of Attribute Syntaxes

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Indicates an optional definition that describes the attribute syntax. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition.
Used by Attributes	Indicates any attributes that use the corresponding attribute syntax.

Viewing Attribute Syntaxes

You can view the attribute syntaxes on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the attribute syntaxes on your Directory Server instance. It shows the properties that are allowed within the attribute syntax.

To View Attribute Syntaxes Over LDAP

- Use `ldapsearch` to view the Directory Server's published list of attribute syntaxes using the multi-valued operational attribute, `ldapSyntaxes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema \  
--searchScope base "(objectclass=*)" ldapSyntaxes
```

Using the Schema Editor Utilities

The Schema Editor provides a **Schema Utilities** tab that enables importing new schema elements from a file and to checking schema compliance. If importing a schema file, the system automatically checks for compliance prior to the import. If the definition does not meet schema compliance, the system will display an error message. However, it is good practice to first check if your file is compliant with your schema prior to importing it.

To Check Schema Compliance Using the Schema Editor

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the Schema Editor, click the **Schema Utilities** tab.
4. Click **Import Schema Elements** to read in an LDIF file, or copy-and-paste a new schema definition, and then click **Validate Entries**. If there is a problem, an error will be generated.

Modifying a Schema Definition

The Directory Server only allows schema definitions that are read-write to be edited. Schema elements indicated by the **Modifiable** column in the Schema Editor's tables can be modified.

To Modify a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is running.
2. On the main menu, click **Schema**.
3. On the Schema Editor, click the **Object Classes** tab.
4. Select the object class that you want to modify, and then click **Actions -> Edit**. The Edit dialog box appears.

5. Make your changes, and then click **OK**.

Deleting a Schema Definition

The Directory Server only allows schema definitions that are read-write to be deleted. In general, those schema definitions in the **Custom** folder of the Schema Editor can be removed from the system. You should make sure that the schema element is not currently in use.

To Delete a Schema Definition

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab.
4. Select the object class that you want to remove, and then click **Actions -> Delete**.
5. On the **Confirmation** dialog, click **Yes** if you are sure that you want to delete the schema element.

Schema Checking

The Ping Identity Directory Server provides full support for parsing all schema elements and provides access to all of its components. By default, the Directory Server enables schema checking for all operations, especially when importing data to the server or when modifying entries using the `ldapmodify` tool. Any schema violations will generate an error message to standard output.

To View the Schema Checking Properties

- Use `dsconfig` to view the schema checking property.

```
$ bin/dsconfig get-global-configuration-prop \  
--property check-schema
```

To Disable Schema Checking

Although not recommended, you can use the `dsconfig` tool to disable the schema checking. This feature only applies to public backends. Schema checking is enforced on private backends, such as changes to the Configuration, Schema, Task, and others. An admin action alert will be generated when attempting to disable schema checking using `dsconfig` interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

Run the `dsconfig` command and specify the `set-global-configuration-prop` subcommand to disable the `check-schema` property.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false
```

The system generates an admin action alert providing alternate options to disabling schema checking. Press **Enter** to continue the process or following one of the suggested tasks:

One or more configuration property changes require administrative action or confirmation/notification.

Those properties include:

- * `check-schema`: Schema checking should only be disabled as a last resort since disabling schema checking harms performance and can lead to unexpected behavior in the server as well as the applications that access it. There are less severe options for addressing schema issues:
 1. Update the data to conform to the server schema.
 2. Modify the server schema to conform to the data. Contact support before modifying the server's default schema.
 3. Change the `single-structural-objectclass-behavior` property to allow entries to have no structural object class or multiple structural object classes.
 4. Change the `invalid-attribute-syntax-behavior` property to allow attribute values to violate their attribute syntax.
 5. Change the `allow-zero-length-values` property of the Directory String Attribute Syntax configuration to allow attributes with this syntax to have a zero length value.

Continue? Choose 'no' to return to the previous step (yes / no) [yes]:

Managing Matching Rule Uses

Matching Rule Use definitions map certain attribute types with a matching rule definition for extensible match filters. Extensible match filters allows clients to search using DN components, for example, `(ou:dn:=engineering)` or using an OID number, for example, `(cn:1.2.3.4:=Sam Carter)`. The matching rule use attribute publishes those attribute types and matching rule combinations, which can be used in extensible match assertions.

Typically, you define a matching rule use that is not normally specified in the attribute type definition. You can create new matching rule uses from the existing schema definitions by adding a custom schema file in the `<server-root>/config/schema` directory.

Matching Rule Use Definitions

Matching Rule Use can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.4 as follows:

```
MatchingRuleUseDescription = "(" wsp
numericoid                  ; Object identifier
[ sp "NAME" sp qdescrs ]    ; Short name descriptor
[ sp "DESC" sp qdstring ]   ; Description
```

```
[ sp "OBSOLETE" ]           ; Specifies if the rule use is inactive
sp "APPLIES" sp oid         ; Attribute types
extensions wsp ")"         ; Extensions followed by a white space and ")"
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                          ; the schema element is marked as read-only in the
                          ; server configuration.
```

To View Matching Rule Uses

A matching rule use lists the attribute types that are suitable for use with an `extensibleMatch` search filter.

- Use `ldapsearch` to view the Directory Server's published list of matching rule uses using the multi-valued operational attribute, `matchingRuleUse`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRuleUse
```

Managing DIT Content Rules

DIT Content Rules provide a way to precisely define what attributes may be present in an entry, based on its structural object class, without specifically creating a new object class definition. The DIT content rules can define the mandatory and optional attributes that entries contain, the set of auxiliary object classes that entries may be part of, and any optional attributes from the structural and auxiliary object classes that are prohibited from being present in the entries.

DIT Content Rule Definitions

DIT Content Rules can be specified with existing schema components and do not require additional code for its implementation. On the Ping Identity Directory Server, only one DIT Content Rule may be defined for an entry in the structural object class.

The formal specification for attribute types is provided in RFC 4512, section 4.1.6 as follows:

```
DITContentRuleDescription = "(" wsp
numericoid                 ; Object identifier of the structural object class the rule
applies to
[ sp "NAME" sp qdescrs ]   ; Short name descriptor
[ sp "DESC" sp qdstring ]  ; Description
[ sp "OBSOLETE" ]         ; Specifies if the rule is inactive
[ sp "AUX" sp oids ]       ; List of allowed auxiliary object classes
[ sp "MUST" sp oids ]      ; List of required attributes
[ sp "MAY" sp oids ]       ; List of allowed attributes in the entry
[ sp "NOT" sp oids ]       ; List of prohibited attributes in the entry
extensions wsp ")"        ; Extensions followed by a white space and ")"
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
```

```
"X-ORIGIN" / ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" / ; Specifies which schema file contains the definition
"X-READ-ONLY" ; True or False. Specifies if the file that contains
; the schema element is marked as read-only in
; the server configuration.
```

To View DIT Content Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITContentRules`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
"(objectclass=*)" dITContentRules
```

Managing Name Forms

Name Forms define how entries can be named based on their structural object class. Specifically, name forms specify the structural object class to be named, as well as the mutually-exclusive set of required and allowed attributes to form the Relative Distinguished Names (RDNs) of the entries. Each structural object class may be associated with at most one name form definition.

Name Form Definitions

Name Forms can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.2 as follows:

```
NameFormDescription = "(" wsp
numericoid           ; object identifier
[ sp "NAME" sp qdescrs ] ; short name descriptor
[ sp "DESC" sp qdstring ] ; description
[ sp "OBSOLETE" ]     ; not active
sp "OC" sp oid        ; structural object class
sp "MUST" SP oids     ; attribute types
[ sp "MAY" sp oids ]  ; attribute types
extensions wsp ")"    ; extensions followed by a white space and ")"
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" / ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" / ; Specifies which schema file contains the definition
"X-READ-ONLY" ; True or False. Specifies if the file that contains
; the schema element is marked as read-only in
; the server configuration.
```

To View Name Forms

- Use `ldapsearch` to view a multi-valued operational attribute `nameForms`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.


```
$ bin/ldapsearch --baseDN cn=schema --searchScope base "(objectclass=*)" nameForms
```

Managing DIT Structure Rules

DIT Structure Rules define which entries may be superior or subordinate to other entries in the DIT. Together with name forms, DIT Structure Rules determine how RDNs are added together to make up distinguished names (DNs). Because DITs do not have a global standard and are specific to a company's implementation, each DIT structure rule associates a name form with an object class and specifies each structure rule with an integer rule identifier, instead of an OID number. The identifier defines its relationship, either superior or subordinate, to another object class. If no superior rules are specified, then the DIT structure rule applies to the root of the subtree.

DIT Structure Rule Definition

DIT Structure Rules can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.1 as follows:

```
DITStructureRuleDescription = "(" wsp
ruleid                       ; object identifier
[ sp "NAME" sp qdescrs ]    ; short name descriptor
[ sp "DESC" sp qdstring ]   ; description
[ sp "OBSOLETE" ]           ; specifies if the rule is inactive
sp "FORM" sp oid            ; OID or name form with which the rule is associated
[ sp "SUP" ruleids ]        ; Superior rule IDs
extensions wsp ")"          ; extensions followed by a white space and ")"
```

The following extensions are specific to the Ping Identity Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /                ; Specifies where the rule is defined
"X-SCHEMA-FILE" /          ; Specifies which schema file contains the definition
"X-READ-ONLY"               ; True or False. Specifies if the file that contains
                           ; the schema element is marked as read-only in
                           ; the server configuration.
```

To View DIT Structure Rules

- Use `ldapsearch` to view a multi-valued operational attribute `dITStructureRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
"(objectclass=*)" dITStructureRules
```

Managing JSON Attribute Values

The Ping Identity Directory Server supports a JSON object attribute syntax, which can be used for attribute types whose values are JSON objects. The syntax requires that each value of this type is a valid JSON object. The following is an example schema definition:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( jsonAttr1-OID NAME 'jsonAttr1' DESC 'test json attribute support'
EQUALITY jsonObjectExactMatch SYNTAX 1.3.6.1.4.1.30221.2.3.4 USAGE userApplications )
objectClasses: ( jsonObjectClass-OID NAME 'jsonObjectClass' AUXILIARY MAY jsonAttr1 )
```



Note: The EQUALITY matching rule should always be specified as `jsonObjectExactMatch` in the schema definition. Using the `jsonObjectFilterExtensibleMatch` is not valid in this case.

Two matching rules are provided to filter the JSON object syntax: `jsonObjectExactMatch` and `jsonObjectFilterExtensibleMatch`.

The `jsonObjectExactMatch` equality matching rule is used in evaluating equality filters in search operations, and for matching performed against JSON object attributes for add, compare, and modify operations. It determines whether two values are logically-equivalent JSON objects. The field names used in both objects must match exactly (although fields may appear in different orders). The values of each field must have the same data types. String values are case-insensitive. The order of elements in arrays is considered significant. Substring or approximate matching is not supported.

The `jsonObjectFilterExtensibleMatch` matching rule can perform more powerful matching against JSON objects. The assertion values for these extensible matching filters should be JSON objects that express the constraints for the matching. These JSON object filters are described in detail in the Javadoc documentation for the LDAP SDK for Java. Although the LDAP SDK can facilitate searches with this matching rule, these searches can be issued through any LDAP client API that supports extensible matching.

The following are example searches using the `jsonObjectFilterExtensibleMatch` rule with available filter types.

Equals field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com -D "cn=Directory Manager" -w password
'(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" : "equals", "field" :
["stuff", "onetype", "name"], "value" : "John Doe" })'
```

Contains field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com -D "cn=Directory Manager" -w password
'(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" : "containsField",
"field" : "age", "expectedType" : "number" })'
```

Greater than field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com -D "cn=Directory Manager" -w password
'(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" : "greaterThan", "field" :
"age", "value" : 26, "allowEquals" : true})'
```

Indexing is supported only for the `jsonObjectExactMatch` matching rule. If possible, non-baseObject searches that use the `jsonObjectFilterExtensibleMatch` matching rule should be wrapped in an LDAP `AND` filter that also contains one or more indexed components so that the search can be processed efficiently.

Configuring JSON Attribute Constraints

The Ping Identity Directory Server can define a number of constraints for the fields included in JSON objects stored in values of a specified attribute type. Constraints that can be placed on a JSON field include:

- Requiring values of the field to have a specified data type.
- Indicating whether the field is required or optional.
- Indicating whether the field can have multiple values in an array. If a field is permitted to have array values, restrictions can also be placed on the number of elements that can be present in the array.
- Indicating whether the field can have a value that is the null primitive as an alternative to values of the indicated data type.
- Restricting values of string fields to a predefined set of values, that match a given regular expression, or a specified length.
- Restricting values of numeric fields with upper and lower bounds.

Any existing data that doesn't match newly-defined JSON constraints can still be decoded and managed by the server. Only new entries are subject to the new constraints. Attempts to alter existing entries with non-compliant JSON objects may require fixing those objects to make them conform to the new constraints.

The two global configuration properties that define schema constraints for JSON objects are `create-json-attribute-constraints` and `create-json-field-constraints` in `dsconfig`. In `dsconfig` interactive under advanced settings, the menu options are JSON Attribute Constraints and JSON Field Constraints. Configuration properties for each include:

- **attribute-type.** The name or object identifier of the attribute type with which the definition is associated. This attribute type must have the JSON object syntax. This property will be the naming attribute for the configuration entry.
- **allow-unnamed-fields.** A boolean value that indicates whether JSON objects, used as the values of attributes of the associated type, can include fields that are not referenced in the `attribute-value-constraints` object. If this is `false`, JSON objects will only be permitted to have the defined fields. If this is `true` (which is the default behavior), JSON objects are permitted to have fields that are not referenced, and no constraints are imposed on those fields.

Unless a schema definition is configured with `allow-unnamed-fields` set to `false`, it is only necessary to include information about fields whose values should be indexed or tokenized. However, it may be desirable to define other fields that are expected in order to ensure that clients will not be permitted to store invalid values. See the [Working with JSON Indexes](#) for information about indexing JSON attributes.

As with standard LDAP schema, JSON schema constraints are enforced for any changes made after the constraints are defined. If there are already JSON values in the data before a JSON schema is defined for that attribute type (or before changes are made), values that already exist may violate those constraints. JSON schema constraints will also be enforced for data provided in an LDIF import, so that entries containing JSON objects that violate these constraints will be rejected.

Table 41: JSON Field Constraints

Property	Description
<code>field</code>	Specifies the path to the target field as a string, with periods to separate levels of hierarchy. If any field name in the hierarchy itself includes a period, that period should be escaped with a backslash.
<code>value-type</code>	Specifies the expected data type for the target field. Values can include: <ul style="list-style-type: none"> <code>any</code>. The target field can have any value. <code>boolean</code>. The target field must have a value of <code>true</code> or <code>false</code>. <code>integer</code>. The target field must have a number that can be exactly represented as an integer. <code>null</code>. The target field must have a value of <code>null</code>. <code>number</code>. The target field must have a value that represents a valid JSON number. <code>object</code>. The target field must have a value that represents a valid JSON object. The <code>allowed-fields</code> array may contain additional elements that define constraints for the fields that may be present in the object. <code>string</code>. The target field must have a value that represents a valid JSON string.
<code>is-required</code>	Specifies whether the target field is required to be present. If it is present, its value must be either <code>true</code> or <code>false</code> . If it is absent, a default of <code>false</code> is assumed.
<code>is-array</code>	Indicates whether the target field can be an array. If the value can be an array, all of the elements of the array must be of the type specified in the <code>value-type</code> field. It can be present with a value that is one of the single strings listed below. If it is absent, a value of <code>prohibited</code> is assumed. Values include: <ul style="list-style-type: none"> <code>required</code>. The target field must be an array with zero or more values of the specified type, and must not be a single value of the specified type. <code>optional</code>. The target field may be an array with zero or more values of the specified type, or it may be a single value of the specified type. <code>prohibited</code>. The target field must not be an array and may only be a single value of the specified type.
<code>allow-null-value</code>	Specifies whether the target field can have a value of <code>null</code> as an alternative to the specified <code>value-type</code> . If present, its value must be either <code>true</code> or <code>false</code> . If it is absent, a default of <code>false</code> is assumed. This has no effect if the <code>value-type</code> is <code>null</code> .
<code>allow-empty-object</code>	If the value of the target field is a JSON object (or an array of JSON objects), specifies whether empty objects are permitted. If present, the value must be either <code>true</code> (the object may have zero or more fields) or <code>false</code> (the object may have one or more fields). If it is absent, a default of <code>false</code> is assumed.
<code>index-values</code>	Specifies whether values of the target field should be indexed in backends. If present, the value must be either <code>true</code> (the field should be indexed) or <code>false</code> (the field should

Property	Description
	not be indexed). If it is absent, a default of <code>false</code> is assumed. If <code>true</code> , the <code>value-type</code> must be <code>boolean</code> , <code>integer</code> , <code>null</code> , or <code>string</code> .
<code>index-entry-limit</code>	Specifies the maximum number of entries in which a particular value may appear before the entry ID list for that value is no longer maintained. If present, the value must be an integer greater than or equal to one. If it is absent, the server will use the default index entry limit for the associated backend. This is only applicable if <code>index-values</code> is <code>true</code> .
<code>prime-index</code>	Specifies whether backends should prime the contents of the JSON index database into memory when they are opened. This is ignored if the backend's <code>prime-all-indexes</code> property has a value of <code>true</code> .
<code>cache-mode</code>	Specifies the cache mode to use for the contents of the JSON index database. If the value is not specified, the backend's default cache mode will be used. If a cache mode of <code>cache-keys-only</code> is configured, priming will only load the internal nodes into memory for the index. If <code>no-caching</code> is configured, no priming is performed for the index.
<code>tokenize-values</code>	Specifies whether values of the target field should be tokenized in backends. If present, the value is either <code>true</code> (field values should be tokenized) or <code>false</code> . If it is absent, a value of <code>false</code> is assumed. If <code>true</code> , the <code>value-type</code> must be <code>string</code> .
<code>allowed-values</code>	Specifies an explicit set of values that the target field is permitted to have. If present, the value must be an array of strings. Any attempt to use a value not in this array for the associated field is rejected. This can only be used with a <code>value-type</code> of <code>string</code> . If not present, any value can be used as long as it satisfies all other defined constraints.
<code>allowed-value-regular-expression</code>	Specifies a regular expression that the target field will be required to match. If present, the value must be a single string or an array of strings representing valid regular expressions (which may require escaping to represent in JSON). Any attempt to use a value that does not match one of the provided regular expressions will be rejected. This can only be used with a <code>value-type</code> of <code>string</code> . If not present, values are not required to match any regular expression.
<code>minimum-numeric-value</code>	Specifies the lower bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is less than this number is rejected. This can only be used with a <code>value-type</code> of <code>integer</code> or <code>number</code> . If not present, no minimum numeric value applies.
<code>maximum-numeric-value</code>	Specifies the upper bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is more than this number is rejected. This can only be used with a <code>value-type</code> of <code>integer</code> or <code>number</code> . If not present, no maximum numeric value applies.
<code>minimum-value-length</code>	Specifies the minimum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with fewer characters than this number is rejected. This can only be used with a <code>value-type</code> of <code>string</code> . If not present, no minimum length applies.
<code>maximum-value-length</code>	Specifies the maximum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with more characters than this number is rejected. This can only be used with a <code>value-type</code> of <code>string</code> . If not present, no maximum length applies.
<code>minimum-value-count</code>	Specifies the minimum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with fewer elements is rejected. This can only be used with an <code>is-array</code> value of <code>required</code> or <code>optional</code> . If not present, no minimum array value count applies.
<code>maximum-value-count</code>	Specifies the maximum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with more elements is rejected. This can only be used with an <code>is-array</code> value of <code>required</code> or <code>optional</code> . If not present, no maximum array value count applies.



Note: When writing JSON objects in a local DB backend, field names and JSON primitive values of `null`, `true`, and `false` are always tokenized. Integers are either tokenized or compacted using their two's complement representation (other numbers are stored using string representations). Array and object sizes are compacted, and their contents are compacted based

on their data types. String values are tokenized that match a recognizable format, including:

- Dates and times in common generalized time and ISO 8601 formats.
 - UUIDs in which the alphabetic characters are either all uppercase or all lowercase.
 - Strings of at least 12 bytes that are a valid base64 encoding.
 - Strings of at least 6 bytes that are a valid hexadecimal encoding, in which the alphabetic characters are either all uppercase or all lowercase.
-

To Add Constraints to JSON Attributes

- Use the `dsconfig` tool (interactive or command line) to create and configure JSON attribute constraints. The following is a sample command:

```
$ bin/dsconfig create-json-attribute-constraints \  
  --attribute-type appjson \  
  --set enabled:true \  
  --set allow-unnamed-fields:false
```

```
$ bin/dsconfig create-json-field-constraints \  
  --attribute-type appjson \  
  --json-field email.verified \  
  --set value-type:boolean \  
  --set is-required:true \  
  --set index-values:true \  
  --set tokenize-values:false \  
  --set allow-null-value:true
```

```
$ bin/dsconfig create-json-field-constraints \  
  --attribute-type appjson \  
  --json-field email.type \  
  --set value-type:string \  
  --set is-required:true \  
  --set index-values:true \  
  --set tokenize-values:true \  
  --set allowed-value:home \  
  --set allowed-value:other \  
  --set allowed-value:work \  
  --set allow-null-value:false \  
  --set minimum-value-length:1
```

```
$ bin/dsconfig create-json-field-constraints \  
  --attribute-type appjson \  
  --json-field email.value \  
  --set value-type:string \  
  --set is-required:true \  
  --set index-values:true \  
  --set tokenize-values:true \  
  --set prime-index:true \  
  --set allow-null-value:true \  
  --set maximum-value-length:256 \  
  --set minimum-value-length:1 \  
  --set allowed-value-regular-expression:[-_\+\.\w\d]+@\w+\.\w{2,5}
```

Chapter

18 Managing Password Policies

The Ping Identity Directory Server provides a flexible password policy system to assign, manage, or remove password policies for root and non-root users. The password policy contains configurable properties for password expiration, failed login attempts, account lockout and other aspects of password and account maintenance on the Directory Server. The Directory Server also provides a global configuration option and a per-user privilege feature that disables parts of the password policy evaluation for production environments that do not require a password policy.

This chapter presents the following topics:

Topics:

- [Viewing Password Policies](#)
- [About the Password Policy Properties](#)
- [Modifying an Existing Password Policy](#)
- [Creating a New Password Policy](#)
- [Deleting a Password Policy](#)
- [Modifying a User's Password](#)
- [Enabling YubiKey Authentication](#)
- [Enabling Social Login](#)
- [Managing User Accounts](#)
- [Disabling Password Policy Evaluation](#)
- [Managing Password Validators](#)

Viewing Password Policies

Password policies enforce a set of rules that ensure that access to data is not compromised through negligent password practices. The Ping Identity Directory Server provides mechanisms to create and maintain password policies that determine whether passwords should expire, whether users are allowed to modify their own passwords, or whether too many failed authentication attempts should result in an account lockout. Many other options are available to fully configure a password policy for your PingData Platform system.

The Directory Server provides three out-of-the-box password policies that can be applied to your entries or as templates for configuring customized policies. The Default Password Policy is automatically applied to all users although it is possible to use an alternate password policy on a per-user basis. The Root Password Policy is enforced for the default root user, which uses a stronger password storage scheme (PBKDF2 rather than the salted 256-bit SHA-2 scheme) and also requires that a root user provide his or her current password to select a new password.

The Secure Password Policy provides a more secure option than the default policy that makes use of a number of features, including password expiration, account lockout, last login time and last login IP address tracking, password history, and a number of password validators.

Caution:



Using the Secure Password policy as-is may notably increase write load in the server by requiring updates to password policy state attributes in user entries and/or requiring users to change passwords more frequently. In environments in which write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it may be useful to consider whether the policy should be updated to reduce the number of entry updates that may be required.

To View Password Policies

- You can view the list of password policies configured on the Directory Server using the `dsconfig` tool, in either interactive or non-interactive mode, or the Administrative Console. The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode:

```
$ bin/dsconfig list-password-policies
```

```

Password Policy      : Type : password-attribute : default-password-storage-scheme
-----:-----:-----:-----
Default Password Policy : generic : userpassword      : Salted 256-bit SHA-2
Root Password Policy  : generic : userpassword      : PBKDF2
Secure Password Policy : generic : userpassword      : PBKDF2
    
```


To View a Specific Password Policy

- Use `dsconfig` or the Administrative Console to view a specific password policy. In this example, view the Default Password Policy that applies to all uses for which no specific policy has been configured.

```
$ bin/dsconfig get-password-policy-prop \
  --policy-name "Default Password Policy"
```

```
Property                               : Value(s)
-----
description                             : -
password-attribute                       : userpassword
default-password-storage-scheme          : Salted SHA-1
deprecated-password-storage-scheme       : -
password-validator                       : -
account-status-notification-handler       : -
allow-user-password-changes              : true
password-change-requires-current-password : false
force-change-on-add                      : false
force-change-on-reset                    : false
password-generator                       : Random Password Generator
require-secure-authentication             : false
require-secure-password-changes          : false
min-password-age                         : 0s
max-password-age                         : 0s
max-password-reset-age                   : 0s
password-expiration-warning-interval      : 5d
expire-passwords-without-warning         : false
allow-expired-password-changes           : false
grace-login-count                        : 0s
lockout-failure-count                    : 0s
lockout-duration                         : 0s
lockout-failure-expiration-interval       : 0s
require-change-by-time                   : -
last-login-time-attribute                 : ds-pwp-last-login-time
last-login-time-format                   : -
previous-last-login-time-format           : -
idle-lockout-interval                    : 0s
password-history-count                   : 0s
password-history-duration                 : 0s
```

About the Password Policy Properties

The Directory Server provides a number of configurable properties that can be used to control password policy behavior.



Note: To view a description of each of the password policy properties, see the *PingData Directory Server Configuration Reference* that is bundled with the Ping Identity Directory Server.

Some of the most notable properties include:

- **allow-user-password-changes.** Specifies whether users can change their own passwords. If a user attempts to change his/her own password, then the server will consult this property for the user's password policy, and will also ensure that the access control handler allows the user to modify the configured password attribute.

- **default-password-storage-scheme.** Specifies the names of the password storage schemes that are used to encode clear-text passwords for this password policy.
- **enable-debug.** When enabled, is used to debug password policy interaction. This property should be used in addition to the server's debug framework with a relevant debug target.
- **force-change-on-add.** Specifies whether users are required to change their passwords upon first authenticating to the Directory Server after their account has been created.
- **force-change-on-reset.** Specifies whether users are required to change their passwords after they have been reset by an administrator. An administrator is a user who has the `password-reset` privilege and the appropriate access control instruction to allow modification of other users' passwords.
- **idle-lockout-interval.** Specifies the maximum length of time that an account may remain idle (the associated user does not authenticate to the server) before that user is locked out. For accounts that do not have a last login time value, the password changed time or the account creation time will be used. If that information is not available, then the user will not be allowed to authenticate. It is strongly recommended that the server be allowed to run for a period of time with last login time tracking enabled (i.e., values for both `last-login-time-attribute` and `last-login-time-format` properties) to ensure that users have a last login time before enabling idle account lockout.
- **lockout-duration.** Specifies the length of time that an account is locked after too many authentication failures. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. A value of 0 seconds indicates that the account must remain locked until an administrator resets the password.
- **lockout-failure-count.** Specifies the maximum number of times that a user may be allowed to attempt to bind with the wrong password before that user's account becomes locked either temporarily (in which case the account will automatically be unlocked after a configurable length of time) or permanently (in which case an administrator must reset the user's password before the account may be used again). For example, if the value is set to 3, the user is locked out after three failed attempts, even if a fourth attempt is made with the correct password.
- **max-password-age.** Specifies the maximum length of time that a user can continue to use the same password before he or she is required to choose a new password. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w). A minimum length of time can also be specified before the user is allowed to change the password.
- **password-change-requires-current-password.** Specifies whether users must include their current password when changing their password. This applies for both password changes made with the password modify extended operation as well as simple modify operations targeting the password attribute. In the latter case, if the current password is required then the password modification must remove the current value and add the desired new value (providing both the current and new passwords in the clear rather than using encoded representations).
- **password-expiration-warning-interval.** Specifies the length of time before a user's password expires that he or she receives notification about the upcoming expiration (either through the password policy or password expiring response controls). The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).

- **password-retirement-behavior.** Specifies the behavior of a password that is allowed a retirement period before becoming invalid. This setting may be used by application service accounts that require a transition period while updating passwords. This is disabled by default.
- **password-validator.** Specifies the names of the password validators that are used with the associated password storage scheme. The password validators are invoked when a user attempts to provide a new password, to determine whether the new password is acceptable.
- **require-secure-authentication.** Indicates whether users with the associated password policy are required to authenticate in a secure manner. This might mean either using a secure communication channel between the client and the server, or using a SASL mechanism that does not expose the credentials.
- **require-secure-password-changes.** Indicates whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials.

Modifying an Existing Password Policy

You can modify an existing password policy to suit your company's requirements.

To Modify an Existing Password Policy

- Use `dsconfig` (in interactive or non-interactive mode) or the Administrative Console to modify the configuration for any defined password policy. The following example sets some of the properties presented in the previous section for the default password policy using `dsconfig` in non-interactive mode:

```
$ bin/dsconfig set-password-policy-prop \
--policy-name "Default Password Policy" \
--set "max-password-age:90 days" \
--set "password-expiration-warning-interval:14 days" \
--set "lockout-failure-count:3" \
--set "password-history-count:6"
```

Creating a New Password Policy

You can create any number of password policies in the Directory Server using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console.

To Create a New Password Policy

- Use `dsconfig` (in interactive or non-interactive mode) or the Administrative Console to create a new password policy. The following example demonstrates the process for creating a new policy using `dsconfig` in non-interactive mode.

```
$ bin/dsconfig create-password-policy \
```

```
--policy-name "Demo Password Policy" \
--set "password-attribute:userpassword" \
--set "default-password-storage-scheme:Salted 256-bit SHA-2" \
--set "force-change-on-add:true" \
--set "force-change-on-reset:true" \
--set "password-expiration-warning-interval:2 weeks" \
--set "max-password-age:90 days" \
--set "lockout-duration:24 hours" \
--set "lockout-failure-count:3" \
--set "password-change-requires-current-password:true"
```

To Assign a Password Policy to an Individual Account

To indicate that a user should be subject to a particular password policy (rather than automatically inheriting the default policy), include the `ds-pwp-password-policy-dn` operational attribute in that user's entry with a value equal to the DN of the desired password policy for that user. This attribute can be explicitly included in a user's entry, or it can be generated by a virtual attribute, which makes it easy to apply a custom password policy to a set of users based on a flexible set of criteria.

1. Create a file (`assign.ldif`) with the following contents:

```
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password Policies,cn=config
```

2. Use `ldapmodify` to apply the modification to the user's entry.

```
$ bin/ldapmodify --filename assign.ldif
```

To Assign a Password Policy Using a Virtual Attribute

It is possible to automatically assign a custom password policy for a set of users using a virtual attribute. The virtual attribute can be configured so that it can use a range of criteria for selecting the entries for which the virtual attribute should appear.

1. Create an LDIF file, which may be used to add a group to the server.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Engineering Managers,ou=groups,dc=example,dc=com
objectClass: groupOfUniqueNames
objectClass: top
cn: Engineering Managers
uniqueMember: uid=user.0,ou=People,dc=example,dc=com ou: groups
```

2. Use `ldapmodify` to add the entries to the server.

```
$ bin/ldapmodify --defaultAdd --filename groups.ldif
```

3. Use `dsconfig` to create a virtual attribute that will add the `ds-pwp-password-policy-dn` attribute with a value of `cn=Demo Password Policy,cn=Password Policies,cn=config` to the entries for all users that are members of the `cn=Engineering Managers,ou=Groups,dc=example,dc=com` group.

```
$ bin/dsconfig create-virtual-attribute \
  --name "Eng Mgrs Password Policy" \
  --type user-defined \
  --set "description:Eng Mgrs Grp PWPolicy" \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "value:cn=Demo Password Policy,cn=Password Policies,cn=config" \
  --set "group-dn:cn=Engineering Managers,ou=Groups,dc=example,dc=com"
```

4. Use `ldapsearch` to verify that a user in the group contains the assigned password policy DN.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
  ds-pwp-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password Policies,cn=config
```

Deleting a Password Policy

You can delete a password policy using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console.

To Delete a Password Policy

Custom password policies may be removed using either the `dsconfig` tool (in interactive or non-interactive mode) or the Administrative Console).

- Run the `dsconfig` command with the `delete-password-policy` subcommand to remove a password policy.

```
$ bin/dsconfig delete-password-policy \
  --policy-name "Demo Password Policy"
```

Modifying a User's Password

There are two primary ways to change user passwords in the Directory Server:

- Perform a modify operation which replaces the value of the password attribute (often `userPassword`). Note that in some configurations, when a user attempts to change his or her own password it may be necessary to perform the modification by removing the password value and adding the desired new value to demonstrate that the user knows the current password value.
- Use the password modify extended operation to change the password. Note that if a user is changing his or her own password, it may be necessary to provide the current password value. The server will allow a new password to be provided (assuming that the new password is acceptable to all configured password validators), or it may automatically generate a new password for the user.

Note that regardless of the mechanism used to change the password, all password values should be provided in clear text rather than pre-encoded, and the user will be required to have sufficient access control rights to update the password attribute in the target user's entry. Further, when one user attempts to change the password for another user, then the requester will be required to have the `password-reset` privilege.

Validating a Password

The requirements that the server will impose for a password change can be displayed to users. The `get password quality requirements extended` operation can be used to retrieve information about the requirements, which can then be sent to an end user before an attempted password change. These requirements can also be used to enable client-side validation, so that any password problems can be identified before it is sent to the server. The password validation details request control can be included in an add or modify request, or a password modify extended request, to identify which validation requirements were not met by the password provided in the request.

Password validators can be configured with user-friendly messages that describe the password requirements, and the messages that should be returned if a proposed password does not satisfy those requirements. The server will generate these messages if they are not provided in the configuration.

Password validator properties include `validator-requirement-description` and `validator-failure` message. The following is a simple password validator configuration that requires passwords to contain a minimum of five characters, and lists custom validator messages:

```
$ dsconfig create-password-validator \  
  --validator-name "Minimum 5 Characters Password Validator" \  
  --type length-based --set enabled:true \  
  --set "validator-requirement-description:The password must contain  
    at least 5 characters." \  
  --set "validator-failure-message:The password did not contain  
    at least 5 characters." \  
  --set min-password-length:5
```

After the password validator is created, it should be assigned to a Password Policy to take affect:

```
$ dsconfig set-password-policy-prop \  
  --policy-name "Default Password Policy" \  
  --set "password-validator:Minimum 5 Characters Password Validator"
```

Retiring a Password

An account password can be retired and rotated out of service, instead of immediately invalidated. This enables a new password to be assigned to an account while keeping the original password valid for a period of time to enable a transition. This is useful for application service accounts that require uninterrupted authentication with the server.

This behavior is disabled by default, but can be enabled in the password policy configuration by setting the `password-retirement-behavior` and `maximum-retired-password-age` properties.

To manually retire an account password or purge a password that has been retired, use the `ldapmodify` and `ldappasswordmodify` commands with options `--retireCurrentPassword`

and `--purgeCurrentPassword`. To use these commands on an account, the password policy that governs the account must have the `password-retirement-behavior` enabled.

To Change a User's Password using the Modify Operation

- Use `ldapmodify` to change a user's password by replacing the `userPassword` attribute.

```
$ bin/ldapmodify
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpw
```

To Change a User's Password using the Password Modify Extended Operation

- Use `ldappasswordmodify` to request that the Password Modify extended operation be used to modify a user's password.

```
$ bin/ldappasswordmodify --authzID dn:uid=jdoe,ou=People,dc=example,dc=com \
--newPassword newpw
```

To Use an Automatically-Generated Password

- Use `ldappasswordmodify` to automatically generate a new password for a user.

```
$ bin/ldappasswordmodify --authzID "u:user.1"
```

```
The LDAP password modify operation was successful
Generated Password: fbi27oqy
```

Enabling YubiKey Authentication

Users can be enabled to authenticate with YubiKey devices (available from Yubico), which generate secure one-time passwords, with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. A YubiKey device generates a different password for every authentication attempt, and that one-time password is sent to a validation service to ensure that it is genuine and has not been used in an earlier authentication attempt. Although it is possible to use this one-time password as the only proof of identity, it is typically combined with a static password as a form of two-factor authentication.

YubiKey authentication requires server configuration and the addition of this capability to a user entry. Configuration of a client ID and API key to use when communicating with the validation service is also required. The API key is a shared secret between the YubiKey validation service and the client that is interacting with it, and is used when generating digital signatures so that both the server and the YubiKey validation service can ensure that the peer server is genuine.

All server and user entry configuration details are available in the *Ping Identity Security Guide*.

Enabling Social Login

Authentication involving credentials that do not reside in, or cannot be forwarded to or validated by the Directory Server (such as social login through Facebook, Google, or Twitter) can be enabled with the the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL mechanism. The bind request will not include any credentials, and authentication with this mechanism will not actually change the state of the underlying client connection. The server will behave as if the bind request included the retain identity request control, regardless of whether or not that control was included.

Bind requests using this mechanism can include any request controls that are permitted with other bind requests. If the externally-processed authentication is successful, the client can include the get password policy state issues request control in the bind request to obtain information about any password policy state issues that can cause the Directory Server authentication attempt to fail. The password policy request control can also be included to obtain certain password policy state warnings and errors, or to look for the password expired/password expiring controls in the bind response.

All server and user entry configuration details are available in the *PingData Security Guide*.

Managing User Accounts

The Directory Server provides a user management utility, the `manage-account` tool, that provides a means to quickly view and manipulate a number of password and account policy properties for a user or group of users.



Note: A disabled account status (for example, `account-is-disabled: true`) is different from an *account lockout* due to password policy. Unlocking a user account requires a *password reset* by an administrator. A disabled account requires the administrator to enable the account; password resets are not involved.

To Return the Password Policy State Information

- Use `manage-account` to get information about the account's password policy.

```
$ bin/manage-account get-all \  
--targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Password Policy DN: cn=Demo Password Policy,cn=Password Policies,cn=config  
Account Is Disabled: false  
Account Expiration Time:  
Seconds Until Account Expiration:  
Password Changed Time: 19700101000000.000Z  
Password Expiration Warned Time:  
Seconds Until Password Expiration: 1209600  
Seconds Until Password Expiration Warning: 0  
Authentication Failure Times:
```



```

Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count: 3
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset: false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count: 0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:

```

To Determine Whether an Account is Disabled

- Use `manage-account` to determine whether a user's account has been disabled.

```

$ bin/manage-account get-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com

```

```
Account Is Disabled: true
```

To Disable an Account

- Use `manage-account` to disable a user's account.

```

$ bin/manage-account set-account-is-disabled \
  --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com

```

```
Account Is Disabled: true
```

To Enable a Disabled Account

- Use `manage-account` to enable a user's account.

```

$ bin/manage-account clear-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com

```

```
Account Is Disabled: false
```

To Assign the Manage-Account Access Privileges to Non-Root Users

Non-root users (e.g., `uid=admin`) with admin right privileges require access control permission to interact with certain password policy operational attributes when using the `manage-account` tool.

For example, the presence of the `ds-pwp-account-disabled` operational attribute in an entry determines that the entry is disabled. If the non-root admin user does not have the access privilege to read or interact with the `ds-pwp-account-disabled` operational attribute, the `manage-account` tool may report that the account is active. An account is considered active if the `ds-pwp-account-disabled` operational attribute does not exist in the entry or if the admin user does not have permission to see it.

Use the following procedure to give access rights to the non-root admin user.

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. Grant the `password-reset` privilege to the account. See step 1 and 6 in the [Configuring Administrators](#) section for more information.
2. Run the `manage-account` tool to view the account status for an account.

```
$ bin/manage-account get-all \
  --targetDN uid=user.0,ou=People,dc=example,dc=com

Password Policy DN: cn=Default Password Policy,cn=Password Policies,cn=config
Account Is Disabled: false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time: 19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset: false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count: 0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

3. Grant access control privileges to the admin account. Here you have two options: give full access rights to all user attributes and specify the `ds-pwp-account-disabled` operational attribute.

```
aci: (targetattr="*" | ds-pwp-account-disabled")
  (version 3.0; acl "Allow user to access all user attributes and the
  ds-pwp-account-disabled operational attribute";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

or give full access to all user and operational attributes:

```
aci: (targetattr="*" | "+")
  (version 3.0; acl "Allow user to access all user and operational
  attributes";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

4. Optional. The Directory Server comes with a default set of global ACIs that include access privileges to the password policy extended operation, which is automatically available to all authenticated users. If your PingData Platform system limits access to these Global ACIs, you can grant the password policy extended operation to all authenticated users (`userdn="ldap:///all";`) or to an admin user as follows:

```
aci: (extop="1.3.6.1.4.1.30221.1.6.1")
  (version 3.0; acl "Allow the password policy extended operation";
  allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

5. Run the `manage-account` tool to disable an account. The following command sets the `account-is-disabled` property to `true` for the `uid=user.0,dc=example,dc=com`.

```
$ bin/manage-account set-account-is-disabled \
  --targetDN uid=user.0,ou=People,dc=example,dc=com \
  --operationValue true
```

```
Account Is Disabled: true
```

- Run the `ldapsearch` tool to view the presence of the `ds-pwp-account-disabled` operational attribute in the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" "+"
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-account-disabled: true
```

Disabling Password Policy Evaluation

The Directory Server provides a global configuration property (`disable-password-policy-evaluation`) that can be used to disable most password policy evaluation processing. This provides a convenience for those production environments that do not require password policy support. If the `disable-password-policy` property is set to `true`, passwords will still be encoded and evaluated, but only account expiration and account disabling will be in effect. All other password policy properties, such as password expiration, lockout, and force change on add or reset, are ignored.

The server also supports the use of a `bypass-pw-policy` privilege, which can be used to skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they will be allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry. Note that this privilege will not have any effect for bind operations.

To Globally Disable Password Policy Evaluation

- Use `dsconfig` to set the `disable-password-policy-evaluation` property globally for the Directory Server.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
--set "disable-password-policy-evaluation:false"
```

To Exempt a User from Password Policy Evaluation

- Use `ldapmodify` to add the `bypass-pw-policy` privilege to a user entry.

```
$ bin/ldapmodify
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

Managing Password Validators

A password validator is a password policy component that is used to determine if a new password is acceptable. A password policy can be configured with any number of password validators. If a password policy is configured with multiple password validators, then all of them must consider a proposed new password acceptable before it will be allowed.

Password Validators

The Ping Identity Directory Server offers a number of types of password validators, including those listed in the following table. It is also possible to use the Server SDK to create custom password validators with whatever constraints are necessary for your environment.

Table 42: Password Validators

Password Validators	Description
Attribute Value	Ensures that the proposed password does not match the value of another attribute in the user's entry. The validator can be configured to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping, and it can also reject values which are substrings of another attribute.
Character Set	Ensures that the proposed password contains a sufficient number of characters from one or more user-defined character sets. For example, the validator can ensure that passwords must have at least one lowercase letter, one uppercase letter, one digit, and one symbol.
Commonly-Used Passwords Dictionary	Ensures that the proposed password is not one of 10,000 commonly used passwords. These are words that are common for attackers to use when trying to access user accounts. The Commonly-Used Passwords validator is invoked by the Secure Password Policy by default. The word list is located in <code><server-root>/config/commonly-used-passwords.txt</code> , and can be used to create a custom validator, but should not be modified.
Dictionary	Ensures that the proposed password is not present in a specified dictionary file, optionally also testing the password with all characters in reverse order. A large dictionary file is provided with the server, but the administrator can supply an alternate dictionary. In this case, then the dictionary must be a plain-text file with one word per line.
Haystack Password Validator	Ensures that the proposed password is secure based on a combination of its length and the types of characters that it contains. For example, a longer password containing only lowercase letters may be stronger than a shorter password containing a mix of uppercase and lowercase letters, numbers, and symbols. This is based on the Gibson Research Corporation Password Haystacks concept.
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters may be specified.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.
Repeated Characters	Ensures that a proposed password does not contain a substring in which the same character is repeated more than a specified number of times (for example, "aaaaa" or "aaabbb"). The validator can be configured to operated in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters (for example, you could define all digits as equivalent, so the proposed password could not contain more than a specified number of consecutive digits).
Similarity-Based Password Validator	Ensures that the proposed new password is not too similar to the current password, using the Levenstein Distance algorithm, which calculates the number of characters that need to be inserted, removed, or replaced to transform one string into another. Note that for this password validator to

Password Validators	Description
	be effective, it is necessary to have access to the user's current password. Therefore, if this password validator is to be enabled, the <code>password-change-requires-current-password</code> attribute in the password policy configuration must also be set to true.
Unique Characters	Ensures that the proposed password contains at least a specified minimum number of unique characters, optionally using case-insensitive validation.

Configuring Password Validators

You can use the `dsconfig` configuration tool or the Administrative Console to configure or modify any password validators. Once you have defined your password validators, you can add them to an existing password policy. The following example procedures show the `dsconfig` non-interactive commands necessary to carry out such tasks. If you use `dsconfig` in interactive command-line mode, you can access the Password Validator menu in the Basic Objects menu. For more details on the password validator properties, see the *Ping Identity Directory Server Configuration Reference*.

To View the List of Defined Password Validators

- Use `dsconfig` to view the set of password validators defined in the Directory Server.

```
$ bin/dsconfig --no-prompt list-password-validators
```

To Configure the Attribute Value Password Validator

- Use `dsconfig` to edit the existing default configuration for the Attribute Value Password Validator. For example, the following change configures the validator to only examine a specified set of attributes..

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Attribute Value" \
  --set match-attribute:cn \
  --set match-attribute:sn \
  --set match-attribute:telephonenumber \
  --set match-attribute:uid
```

- Update an existing password policy to use the Attribute Value Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Attribute Value"
```

- Test the Attribute Value Password Validator by submitting a password that is identical to one of the configured attributes (`cn`, `sn`, `telephonenumber`, `uid`).

```
$ bin/ldappasswordmodify --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword user.0
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks defined in the
server: The provided password was found in another attribute in the user entry
```

To Configure the Character Set Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we change the requirement for special characters making them optional in a password, and add a requirement that at least two digits must be included in the password. Thus, in this example, all newly created passwords must have at least one lowercase letter, one uppercase letter, two digits, and optionally any special characters listed.

```
$ bin/dsconfig set-password-validator-prop \
--validator-name "Character Set" \
--remove character-set:1:0123456789 \
--remove "character-set:1::~~\!@#\$\%^&*()-_+[]{}|\;:,.<>/?" \
--add character-set:2:0123456789 \
--add "character-set:0::~~\!@#\$\%^&*()-_+[]{}|\;:,.<>/?" \
--set allow-unclassified-characters:false
```

2. Update an existing password policy to use the Character Set Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
--policy-name "Default Password Policy" \
--set "password-validator:Character Set"
```

3. Test the Character Set Password Validator by submitting a password that meets the requirements (one lowercase letter, one uppercase letter, two digits). The following example should reject the given password because it does not pass the Character Set Password Validator.

```
$ bin/ldappasswordmodify \
--authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abab1
```

To Configure the Length-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the required minimum number of characters in a password to five.

```
$ bin/dsconfig create-password-validator \
--validator-name "Length-Based Password Validator" \
--set max-password-length:5 --set min-password-length:5
```

2. Update an existing password policy to use the Length-Based Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
--policy-name "Default Password Policy" \
--set "password-validator:Length-Based Password Policy"
```

3. Test the Length-Based Password Validator by submitting a password that has fewer than the minimum number of required characters.

```
$ bin/ldappasswordmodify \
--authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abcd
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks defined in
the server: The provided password is shorter than the minimum required length of
5 characters
```

To Configure the Regular Expression Password Validator

1. Use `dsconfig` to create a Regular Expression password validator. The following password validator checks that the password contains at least one number, one lowercase letter, and one uppercase letter with no restrictions on password length. If the password matches the regular expression, then it will be accepted. When using the following command, remember to include the LDAP/LDAPS connection parameters (host name and port), bind DN, and bind password.

```
$ bin/dsconfig create-password-validator \
  --validator-name "Regular Expression" \
  --type regular-expression --set enabled:true \
  --set "match-pattern:^(\\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-Z])\\w*$" \
  --set match-behavior:require-match
```

2. Update an existing password policy to use the Regular Expression validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Regular Expression"
```

3. Test the Regular Expression Validator by submitting a password that meets the requirements (contains one number, one lowercase letter, and one uppercase letter), then run it again with a password that does not meet these requirements.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
```

```
The LDAP password modify operation was successful
```

4. Try another password. The following password should fail, because no uppercase letter is present.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaa1
```

```
Error Message: The provided new password failed the validation checks
defined in the server: The provided password is not acceptable because it does
not match regular expression pattern '^\\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-Z])\\w*$'
```

To Configure the Repeated Character Password Validator

1. Use `dsconfig` to edit the existing default configuration.

- In this example, we set the maximum consecutive length of any character to 3. For example, the following validator rejects any passwords, such as "baaaa1" or "4eeeb", etc.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Repeated Characters" \
  --set max-consecutive-length:3
```

- Or, you can configure the validator to reject any character from a pre-defined character set that appears more than the specified number of times in a row (2). You can also specify more than one character set. For example, the following validator defines two character sets: [abc] and [123]. It rejects any passwords with more than two consecutive characters

from a character set. Thus, "aaa", "bbb", "ccc", "abc", or "123" and so on fails, but "12a3" is okay.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Repeated Characters" \
  --set character-set:123 --set character-set:abc
```

2. Update an existing password policy to use the Repeated Character Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Repeated Characters"
```

3. Test the Repeated Character Validator by submitting a password that has more than the maximum allowable length of consecutive characters.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword baaaal
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks defined
in the server: The provided password contained too many instances of the same
character appearing consecutively. The maximum number of times the same
character may appear consecutively in a password is 2
```

To Configure the Similarity-Based Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of differences to 2.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Similarity-Based Password Validator" \
  --set min-password-difference:2
```

2. Update an existing password policy to use the Similarity-Based Password Validator. The `password-change-requires-current-password` property must be set to `TRUE`, so that the password policy will ensure that the user's current password is available when that user is choosing a new password.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Similarity-Based Password Validator" \
  --set password-change-requires-current-password:true
```

3. Test the Similarity-Based Password Validator by submitting a password that has fewer than the minimum number of changes (e.g., 2). The `ldappasswordmodify` command requires the `--currentPassword` option when testing the Similarity-Based Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --currentPassword abcde --newPassword abcdd
```

```
The LDAP password modify operation failed with result code 49
```

To Configure the Unique Characters Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we set the minimum number of unique characters that a password is allowed to contain to 3.

```
$ bin/dsconfig set-password-validator-prop \  
  --validator-name "Similarity-Based" --set min-unique-characters:3
```

2. Update an existing password policy to use the Unique Characters Password Validator.

```
$ bin/dsconfig set-password-policy-prop \  
  --policy-name "Default Password Policy" \  
  --set "password-validator:Unique Characters"
```

3. Test the Unique Characters Password Validator by submitting a password that has fewer than the minimum number of unique characters (e.g., 3).

```
$ bin/ldappasswordmodify \  
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \  
  --newPassword aaaaa
```

```
The LDAP password modify operation failed with result code 53  
Error Message: The provided new password failed the validation checks defined  
in the server: The provided password does not contain enough unique characters.  
The minimum number of unique characters that may appear in a user password is 3
```


Chapter

19

Managing Replication

The Ping Identity Directory Server provides a robust replication system to ensure high availability and fast failover in production environments. Write requests can be handled by every server in the topology with the replication component performing immediate synchronization with other members. The replicated server environment ensures that LDAP clients can seamlessly fail over to another server instance.

This chapter presents the architectural overview of replication, detailed configuration steps, ways to monitor replication as well as troubleshooting steps:

Topics:

- [Overview of Replication](#)
- [Replication Versus Synchronization](#)
- [Replication Terminology](#)
- [Replication Architecture](#)
- [Replication Deployment Planning](#)
- [Enabling Replication](#)
- [Deploying a Basic Replication Topology](#)
- [A Deployment with Non-Interactive dsreplication](#)
- [Configuring Assured Replication](#)
- [Managing the Topology](#)
- [Advanced Configuration](#)
- [Modifying the Replication Purge Delay](#)
- [Configuring a Single Listener-Address for the Replication Server](#)
- [Monitoring Replication](#)
- [Replication Best Practices](#)
- [Replication Conflicts](#)
- [Troubleshooting Replication](#)
- [Replication Reference](#)
- [Advanced Topics Reference](#)

Overview of Replication

Replication is a data synchronization mechanism that ensures that updates made to a database are automatically replayed to other servers. Replication improves data availability when unforeseen or planned outages occur, and improves search performance by allowing client requests to be distributed across multiple servers.

By default, all Directory Servers participating in replication are writable, so that LDAP clients can perform updates at any of these Directory Server instances. These updates will be propagated to the other servers automatically in the background and applied in the same order as the updates were entered. The replication process flow is designed to immediately propagate changes to the other replicas in the topology with little or no latency.

The following figure demonstrates the basic flow of replication.

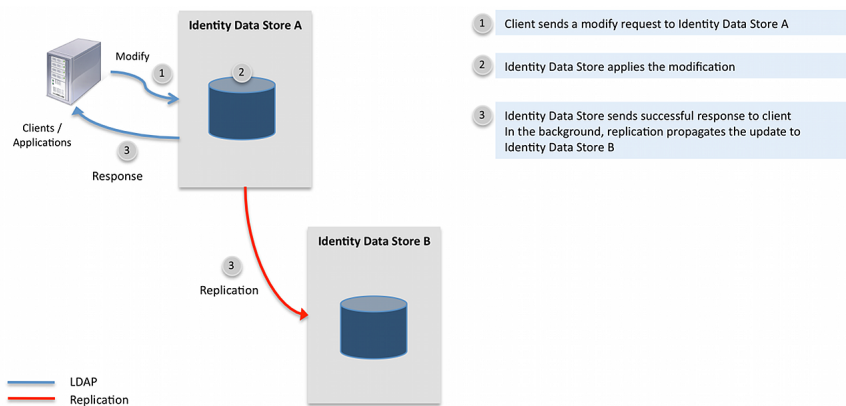


Figure 11: Replication Process Flow

The benefits of replication can be summarized as follows:

- **High-Availability.** Because the data is fully replicated on all other servers in the topology, replication allows participating servers to process all types of client requests. This mitigates any availability issues when a particular server is down due to a planned maintenance or unplanned outage. For those servers that are temporarily unavailable, they will receive updates when they become available again.
- **Improved Search Performance.** Search requests may be directed to any Directory Server participating in replication, which improves search performance over systems that only access single servers. Note, however, that replication does not improve write throughput since updates need to be applied at all servers.
- **WAN Friendly Data Synchronization.** The built-in compression feature in the replication protocol allows efficient propagation of updates over WAN links.

Replication Versus Synchronization

Replication is not a general purpose synchronization facility as it creates replicas with exact copies of the replicated data. Synchronization, on the other hand, can transform data between two different Directory Information Tree (DIT) structures, map attribute types, synchronize subsets of branches and specific object classes. The differences between replication and synchronization are illustrated as follows:

- **Replication cannot Synchronize between Different DIT Structures.** The DN of replicated entries must be the same on all servers. In some situations, it may be desirable to replicate entries with the help of DN mapping that are under different base DNs, but represent the same data, for example `uid=john.doe,ou=people,o=corp` on one server may represent the same user as `uid=john.doe,ou=people,dc=example,dc=com`. This is not supported by replication. Synchronization fully supports this feature.
- **Replication cannot Map Attribute Types or Transform Attribute Values.** In some situations, it may be necessary to map attribute types or transform attribute values when synchronizing data from one server to another. Replication does not support either attribute type mappings or attribute value transformations.
- **Replication does Not Support Fractional Replication.** Replication cannot be configured to replicate a subset of the attribute types from the replicated data set. Synchronization fully supports this feature.
- **Replication does Not Support Sparse Replication.** Replication cannot be configured to replicate entries with a particular object class only. Synchronization fully supports this feature.
- **Replication Requires Full Control of Replicated Data.** When two servers participate in replication, both servers implicitly trust each other using public key cryptography and apply all updates received via replication, which is considered an internal operation. While trust between servers is established between two endpoint servers, synchronization does not require full control of the data. Disparate server system endpoints can be synchronized, such as an Ping Identity Directory Server and a RDBMS database endpoint with each fully in control of its own data.

If replication does not meet your data synchronization requirements, consider using the PingData Data Sync Server, which provides the versatility and robust performance required for most production environments.

Replication Terminology

The following replication terms are used throughout this chapter.

Table 43: Replication Terminology

Term	Description
Assured Replication	For applications that require immediate access to replicated data or require data consistency over response time, administrators can configure <i>assured replication</i> , where data is guaranteed to replicate <i>before</i> the response is returned to the client.
Change Number	A 64-bit number used to consistently order replication updates across the entire topology. It is composed of 3 fields: the timestamp of the update (measured in milliseconds since 00:00:00 UTC, January 1, 1970), the Replica ID, and the Sequence Number.
Conflict	Client updates made at different replicas affecting the same entry may be found in conflict when the updates are replayed at another replica. The Change Number of each update allows most of these conflicts to be resolved automatically. Certain updates, such as adding an entry with different attribute values at two servers simultaneously, result in conflicts that are flagged for required manual resolution.
Eventual Consistency	When not using assured replication, recent updates from LDAP clients are not immediately present at all servers. Out-of-sync data will be eventually synchronized and will be consistent on all servers. The network latency typically controls how long a given update takes to replicate.
Global Administrator	The administrative user with full rights to manage the replication topology. The user is created at the time of replication enable between two non-replicating servers and thereafter copied to newly enabled servers. The replication command-line utility expects the user name of the Global Administrator (by default, admin). The user is stored in <code>cn=admin</code> data.
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify Operation. Historical Data is stored in the <code>ds-sync-hist</code> attribute of the target entry. This information allows replication to resolve conflicts from simultaneous LDAP Modify operations automatically.
Location	The collection of servers that may have similar performance characteristics when accessed from this Directory Server or reside in the same data center or city. A separate location setting may be defined for each data center, such as Austin, London, Chicago, etc. Location settings are used in the selection of the WAN Gateway server.
Modify Conflict	A conflict between two LDAP Modify operations. Conflicts arising from LDAP Modify operations are automatically resolved.
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one Replica.
Replica	The component in the Directory Server that handles interaction with a single replication domain, for example, the <code>dc=example,dc=com</code> replication domain within the <code>userRoot</code> backend.
Replica ID	The unique identifier of a replica that is set automatically in the corresponding Replication Domain configuration entry at each participating server. The replica ID identifies the source of each update in the replication topology.
Replica State	A list of the most recent change numbers of replication updates that have been applied to a replica. There can be at most one change number from each replica in the state. The replica state helps the Replication Server component to determine which updates the Replica has not received yet.
Replication	An automated background process that pushes directory server data changes to all other replicas.

Term	Description
Replication Changelog	<p>A backend maintained by each replication server independently that records updates from each replica. This backend is distinct from the LDAP Changelog and the two should not be confused. The main distinction is as follows:</p> <ul style="list-style-type: none"> • The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at <server-root>/db/changelog. • The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at <server-root>/changelogDB and is not accessible by clients and server extensions.
Replication Domain	The data configured for replication as defined by the base DN. Updates to entries at and below the base DN will be replicated.
Replication Replay	When a replica locally applies the update received via a replication server.
Replication Server	A component within the Directory Server process that is responsible for propagating directory server data changes to and from replicas. Each Directory Server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server set automatically in its configuration object at each server in the replication topology. This identifier is used in the connection management code of the replication servers.
Replication Server State	A list of the most recent change numbers of replication updates that have been processed by a replication server. There can be at most one change number from each replica in the topology. The Replication Server State is used to determine which updates need to be sent to other replication servers. Similarly, the replica can use the Replication Server State to identify the set of updates to send to the replication server.
Sequence Number	A field in the change number that indicates the sequence of the client updates at a particular replica. For every update at the replica, the number is incremented. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.
WAN Gateway	The designated replication server that assumes the WAN Gateway role within a collection of co-located replication servers (i.e., servers that are defined with identical location settings). Replication update messages between servers at different locations are routed through WAN gateways. The WAN Gateway role is assigned automatically by the protocol based on the server's WAN Gateway Priority setting. If the WAN Gateway server is down for any reason, the server with the next highest WAN Gateway Priority will dynamically assume the WAN Gateway role.
WAN Gateway Priority	The configuration setting that determines which replication server assumes the WAN Gateway role. The replication server with the lowest WAN Gateway Priority value in a location assumes the role of the WAN Gateway. The priority values can be set to 0 (never be a gateway), or any value from 1 (highest priority) to 10 (lowest priority).

Replication Architecture

The major elements of replication in the Ping Identity Directory Server are introduced in this section.

Eventual Consistency

Replication is based on an eventual-consistency model, where updates that are propagated through a connected network of servers will eventually be consistent on all servers over a very short period of time. In a typical update operation, a client application updates an entry or group of entries on the Ping Identity Directory Server with an ADD, DELETE, MODIFY, or MODIFY DN operation. After processing the operation, the Directory Server returns an LDAP response, while concurrently propagating the update to the other servers in the replicated topology. This concurrent processing model allows the client to continue submitting update requests without waiting for a replication completion response from the server. Alternatively, assured replication can be configured for specific write requests that requires local or global consistency, across datacenter locations, before a response is returned to the client. For more information, see [Configuring Assured Replication](#) on page 366.

To support this processing model, replication never locks the targeted entries at the other Directory Server instances before an update can be made locally. This means that the replicated Directory Servers may have an inconsistent view of the targeted entry for a very short period of time but will catch up as the propagated changes are applied. The eventual-consistency model also allows clients to complete update operations faster, since clients do not have to wait for replication to propagate the change. The rate of update operations remains the same no matter how many Directory Servers participate in replication.

Replicas and Replication Servers

Each Directory Server has an embedded replication server that is responsible for transmitting updates to other replication servers. There is a separate component, called a replica, for each replicating base DN, such as `cn=schema, dc=example, dc=com`. Each replica connects to the embedded replication server running within the Directory Server JVM process.

When a client application updates an entry on the Directory Server, the replica sends an update message to its embedded replication server. The replication server applies the change to the `replicationChangeLog` backend repository and then sends an update message to the connected replication server on another directory server. The replication server on other directory servers then passes the change to the appropriate replica, which in turn applies the change to its backend after performing conflict resolution. This standard setup is seen in the figure below.

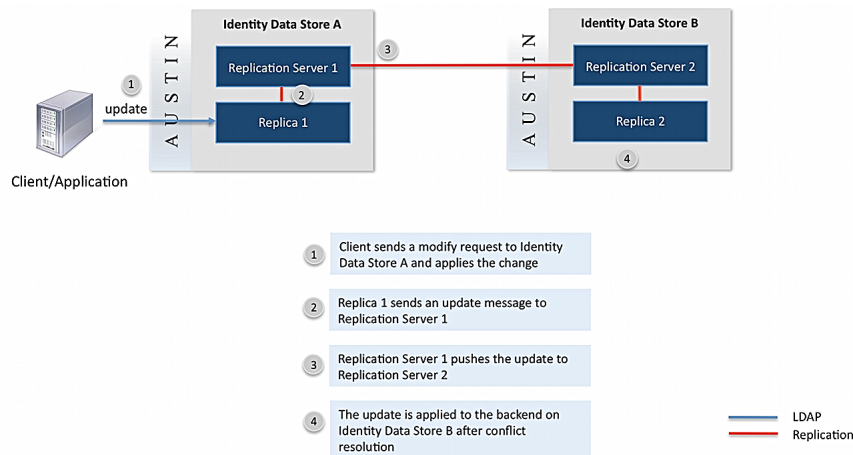


Figure 12: Replicas and Replication Servers

Authentication and Authorization

The authentication in the Replication Protocol is based on public key cryptography using client certificate authentication via TLS. The certificate used for authentication is stored in the `ads-truststore` backend of the Directory Server. During replication setup, the command-line utility distributes public keys to all directory servers to establish trust between the Directory Servers and to enable client authentication via TLS.

The authorization model of replication is simple: once authenticated, the remote Directory Server is fully authorized to exchange replication messages with the local Directory Server. There is no other access control in place.

Logging

The access log messages in the Directory Server indicate if the update was received via replication and includes the corresponding change number. This allows the administrator to track which Directory Server the update originated from.

Replication Deployment Planning

The following should be considered before deploying in a production environment:

- **Minimizing Replication Conflicts.** Attention should be paid to the origin of client write requests to prevent a conflict. If two different clients attempt to create an entry with the same name, DN, at the same time against different servers, the possibility exists that both client requests will succeed. In this case, a conflict alert will be sent by the server. However, understand the client traffic pattern beforehand will minimize these occurrences. The Directory Server's Assured Replication feature and the Ping Identity Directory Proxy Server can both assist in minimizing conflicts.
- **Replication Purge Delay.** Adjust the default one-day replication purge delay, consistently across all servers, to accommodate automatic catchup of changes when a server is offline

for an extended period of time. The replication changes database, stored in `<server-root>/changelogDb`, grows larger as the replication purge delay is increased. A minimum value should be defined.

The rest of this section highlights other topics of consideration.

Location

In multi-site deployments, it is strongly recommended to configure the directory servers with location information using the `dsconfig create-location` command and `dsconfig set-global-configuration-prop` command. The Directory Server cannot determine LAN boundaries automatically, so incorrect location settings can result in undesired WAN communication. By default, replication also compresses all traffic between directory servers in different locations.

We recommend setting up the locations prior to enabling replication. The `dsreplication enable` command prompts for location information if you have forgotten to define the property.

User-Defined LDAP

Directory Servers participating in replication are required to have a uniform user-defined schema. The `dsreplication` command-line utility sets up replication for the schema backend the first time replication is enabled to ensure that future schema changes are propagated to all directory servers.

Disk Space

Replication increases the disk space required for the Directory Server. The Replication Changelog backend keeps changes from all directory servers for 24 hours by default. After this time period, also known as the purge delay, the backend is trimmed automatically.

In addition, within the `userRoot` and other local DN backends, attribute-level changes are recorded for a short period of time in the `ds-sync-hist` attribute of the entry targeted by an LDAP Modify operation. This attribute is used to resolve all conflicts resulting from LDAP Modify operations automatically.

The disk space impact of replication is highly dependent on the rate and size of changes in the replication topology, and the Replication Changelog purge delay.

Memory

Compared to a standalone directory server, replicated directory server instances require slightly more memory. All of the items discussed in the Disk Space section have an impact on the amount of memory the Directory Server is using. The additional replication overhead is typically less than 5%.

Time Synchronization

Even though replication has a built-in mechanism to compensate for the potential clock skew between hosts, it is generally recommended to keep system clocks in sync within the deployment.

Communication Ports

The replication server component in each directory server listens on a TCP/IP port for replication communication (the replication server port). This port, typically 8989, must be accessible from all directory servers participating in replication. The server-to-server communication channel is kept alive using a heartbeat, which occurs every 10 seconds. This traffic will prevent firewalls from closing connections prematurely.

The replication command-line utility (`dsreplication`) requires access to all directory servers participating in replication. This includes the LDAP or LDAPS port of the directory servers.

Keep these communication requirements in mind when configuring firewalls.

Hardware Load Balancers

Replication allows writes to be directed to any directory server in the topology. Distributing write operations in a round-robin fashion, however, may introduce conflicts. In particular, distributing a series of LDAP Add, Delete and Modify DN operations targeting the same DN in quick succession can result in conflicts that require manual intervention. The Assured Replication feature can help prevent conflicts created by the same client application.

If possible, consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single directory server instance at a time. Also, consider using the Ping Identity Directory Proxy Server for load balancing LDAP traffic. The Server Affinity feature in the Directory Proxy Server enables replication-friendly load balancing.

Directory Proxy Server

In addition, to facilitate replication-friendly load balancing, the Ping Identity Directory Proxy Server should be considered in every replication deployment. The Directory Proxy Server can automatically adapt to conditions in backend directory servers using health checks and route traffic accordingly. For example, traffic can be re-routed from directory servers with large backlog of replication updates.

To Display the Server Information for a Replication Deployment

- Run the `dsreplication status` command with the `--displayServerTable` option.

```
$ bin/dsreplication status --displayServerTable
```

```

--- Replication Status for dc=example,dc=com: ---
Server          : Location : Priority (1) : Status
-----:-----:-----:-----:-----
austin-01.example.com:8989 : US      : 1 (*)      : Available
austin-02.example.com:8989 : US      : 2          : Available
london-01.example.com:8989 : UK      : 5          : Available
london-02.example.com:8989 : UK      : 5 (*)      : Available
sydney-01.example.com:8989 : AU      : 5 (*)      : Available
sydney-02.example.com:8989 : AU      : 5          : Available

[1] WAN Gateway Priority. WAN gateways are marked with a *. To minimize
WAN utilization, the server with the WAN Gateway role is the only server
in a location that exchanges updates with remote locations.

--- Replication Status for dc=example,dc=com: Enabled ---
Server          : Location : Entries : Conflict Entries : Backlog :
Recent Change Rate
-----:-----:-----:-----:-----:-----
austin-01.example.com:1389 : US      : 3478174 : 0          : 8       :
333
austin-02.example.com:1389 : US      : 3478174 : 0          : 5       :
345
london-01.example.com:1389 : UK      : 3478174 : 0          : 0       :
349
london-02.example.com:1389 : UK      : 3478174 : 0          : 0       : 5
sydney-01.example.com:1389 : AU      : 3478173 : 0          : 0       :
350
sydney-02.example.com:1389 : AU      : 3478270 : 0          : 30      :
332

```

To Display All Status Information for a Replication Deployment

- Run the `dsreplication status` command with the `--showAll` option. You can also use the `--showAll` option together with the `--displayServerTable` option to see the server table information for the replication topology.

Enabling Replication

Enabling replication between multiple Directory Servers means that any change within the replicated base DN is automatically propagated to all other Directory Servers in the topology. Configuration changes are not replicated, but can be through the use of Server Groups. Therefore, each Directory Server should be configured according to the deployment design.

Overview

To interface with the replication topology, the Directory Server provides a command-line utility, `dsreplication`, that must be used to manage and monitor replication.

Replication setup involves the following basic steps:

- **Set up the servers.** This is the basic installation steps to set up a Directory Server instance.
- **Import or restore data to one server.** After setting up the servers, at least one server should have the target data loaded through `import-ldif` or `restore`.
- **Enable replication between the servers.** Using the `dsreplication` tool, enable replication for each server to be included in the replication topology. The `dsreplication enable`

subcommand should be run $N - 1$ times for a topology of N servers. See [Command Line Interface](#) for more information.

- **Initialize data from source server to every server in the topology.** Run the `dsreplication initialize` subcommand for every target server that needs a copy of the data from the source server.
- **Verifying the replication topology.** Administrators can check the replication status after configuring the topology using the `dsreplication status` tool.

Command Line Interface

Replication topologies are configured and maintained using the `dsreplication` command-line utility, which supports interactive and non-interactive modes. If you are running the server for the first time, we recommend using the `dsreplication` tool in interactive mode.

The `dsreplication` tool has the following format including some important subcommands listed in the section [The dsreplication Command-Line Utility](#):

```
dsreplication {subcommand} {connection parameters}
```

The `dsreplication` tool keeps a history of invocations in the `logs/tools/dsreplication.history` file and keeps a log of up to 10 `dsreplication` sessions in the `logs/tools` directory.

What Happens When You Enable Replication

The `dsreplication enable` subcommand is used to set up replication. The `enable` subcommand carries out the following functions:

- If it does not already exist, the global administrator user is created. The global administrator user has all the rights and privileges to update replication-related configuration objects. Most `dsreplication` subcommands require the global administrator.
- The server instances are registered in the `cn=admin` data tree. The registration includes basic host name, port information as well as the public key used during the replication authentication process.

In case both servers are already participating in replication, the `cn=admin` data tree is merged to retain the server information from existing topologies.

- The embedded replication server is enabled. Servers already in replication will see their replication server configuration updated with the information of the new replication server.
- A replication domain is created for the requested base DNs. In case the first base DN is enabled, the replication domains for two additional base DNs are also enabled: `cn=admin` data and `cn=schema`.
- Initialization for the `cn=admin` data base DN is executed. This will ensure that `cn=admin` data is uniform across the replication topology.

- Initialization for the `cn=schema` base DN is executed. This will ensure that a uniform schema is present in the replication topology.
- Initialization must be performed for the enabled base DNs.

Initialization

Replica initialization transfers of a copy of the backend containing the replication domain to a target server. This should be performed after replication is enabled with the `dsreplication initialize` subcommand. There is no impact on the source server during this process.

- **dsreplication initialize.** The recommended approach for replica initialization. The `dsreplication initialize` subcommand performs the most efficient copy of data needed to initialize one or more replicas on a target server. Any existing data on the target server replica will be lost and the backend containing the base DN will be taken offline on the target server during the initialization.
- **Binary Copy.** The database copy method involves copying database backup files from the source directory server to one or more target servers. The Directory Server provides tools necessary for backing up and restoring backends. Using `server-root/bin/backup`, create a backup of the backend containing the replicated base DN. The backup files then need to be transferred to the target server(s) and restored individually with `server-root/bin/restore`. There are additional considerations when using database copy as the means to initialize a target replica:
 - If encryption is enabled on the servers, then a database `bin/encryption-settings export` then `bin/encryption-settings import` must be performed on the `encryption-settings` backend.
 - When using database copy to initialize a server which has been offline longer than the replication purge delay, the database copy of the `replicationChanges`, `schema`, and `adminRoot` backends are required.

Deploying a Basic Replication Topology

This section describes how to set up a two-server replication topology. The example uses the LDAP and replication server ports 1389 and 8989 respectively.

Table 44: Replica Ports

Host Name	LDAP Port	Replication Port
server1.example.com	1389	8989
server2.example.com	1389	8989

To Deploy a Basic Replication Deployment

1. Install the first directory server with 2000 sample entries.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \ --
rootUserPassword pass --sampleData 10000 --no-prompt
```

2. Install the second directory server either on a separate host or the same host as the first, but with a different LDAP port.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \ --
rootUserPassword pass --no-prompt
```

3. From the first server, run the `bin/dsreplication` command in interactive mode to configure a replication topology:

```
$ bin/dsreplication
```

4. From the Replication Main menu, select the Manage the topology option.

```
>>>> Replication Main Menu
What do you want to do?
    1) Display replication status
    2) Manage the topology (add and remove servers)
    3) Initialize replica data over the network
    4) Initialize replica data manually
    5) Replace existing data on all servers

    q) quit
Enter choice: 2
```

5. From the Manage Replication Topology menu, choose the Enable Replication option.

```
>>>> Manage Replication Topology
Select an operation for more information.
    1) Enable Replication -- add or re-attach a server to the topology
    2) Disable Replication -- permanently remove a running replica from the topology
    3) Remove Defunct Server -- permanently remove an unavailable server from the
        topology
    4) Cleanup Server -- remove replication artifacts from an offline, local server
        (allowing it to be re-added to a topology)

    b) back
    q) quit
Enter choice [b]: 1
```

6. On the Enable Replication menu, read the brief introduction on what will take place during the setup, and then, enter "c" to continue the enable process.
7. Next, enter the LDAP connection parameters for the first of the two replicas that you are configuring. First, enter the host name or IP address of the first server.
8. Next, enter the type of LDAP connection to the first server: 1) LDAP, 2) LDAP with SSL, or 3) LDAP with StartTLS.
9. Type the LDAP listener port for the first replica. If you are a root user, you will see port 389 as the default. Others will see port 1389.

10. Authenticate as a root DN, such as `cn=Directory Manager`. You will be prompted later in the process to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.
11. Repeat steps 7–10 for the second replica.
12. Next, the `dsreplication` tool checks for the base DN on both servers. In order to enable replication, data must be present on at least one of the servers. For this example, press Enter to select the default base DN, `dc=example,dc=com`.

```
Choose one or more available base DNs for which to enable replication:
  1) dc=example,dc=com
  c) cancel
Enter one or more choices separated by commas [1]:
```

Note: If you see the following message:



```
There are no base DNs available to enable replication between the
two servers.
```

In most cases, a base DN was not set up on one of the directory servers or the backend is disabled.

13. Next, the prompt asks if you want to set up entry balancing using the Directory Proxy Server. Press Enter to accept the default (no), since we are not setting up replication in an entry-balanced environment in this scenario. For more information, see the *Ping Identity Directory Proxy Server Administration Guide*.

```
Do you plan to configure entry balancing using the Directory Proxy Server? (yes / no)
[no]:
```

14. Next, enter the replication port for the first replica (default, 8989). The port must be free.
15. If the first server did not have a pre-defined location setting, `dsreplication` will prompt you to enter a location. Press Enter to accept the default (yes) to set up a Location for the first server. Enter the name of the server's location.

```
The first server has not been configured with a location.
Assigning a location to each server in the replication topology reduces
network traffic in multi-site deployments. Would you like to set the
location in the first server? (yes / no) [yes]
The location of the first server: Austin
```

16. Repeat the previous steps for the second directory server. Again, if you did not pre-define a location setting for the second server, you will be prompted to enter this information.
17. At this time, set up the Global Administrator user ID (default is "admin") and a password for this account. The Global Administrator user ID manages the directory servers used in the replication topology.

```
Specify the user ID of the global administrator account
that will be used to manage the PingData Directory Server
instances to be replicated [admin]:
Password for the global administrator:
```



```
Confirm Password:
```

18. Return to the Replication Main Menu and enter the number corresponding to initializing data over the network.
19. On the Initialize Replica Data over the Network menu, select Initialize to initialize data on a single server, and then enter `c` to continue.
20. Next, specify a server in the replication topology. For this example, enter the host name or IP address, LDAP connection type, LDAP port, Global Admin user ID and password of the first server.
21. Next, select the source server that is hosting the data to which the target server will be initialized. For this example, select the first server, since the sample dataset has been loaded onto this server.
22. Next, select the base DN that will be initialized. In most cases, the base DN for the root suffix will be replicated. In this example, `dc=example,dc=com`.
23. Next, select the second server in this example that will have its data initialized, and then enter the Global Admin user ID and password for the target server. Any data present on the target server will be over-written.
24. Press Enter to confirm that you want to initialize data on the target server. When completed, you should see "Base DN initialized successfully."

```
Initializing the contents of a base DN removes all the existing contents of
that base DN. Do you want to remove the contents of the selected base DN's on server
server2.example.com:1389 and replace them with the contents of server
server1.example.com:1389? (yes / no) [yes]:
```

25. On the Initialize Replica Data Over Network menu, enter `b` to back out one level to the main menu. Then, on the Replication Main menu, enter the number to view the replication status.

```
--- Replication Servers: dc=example,dc=com ---
Server          : Location : Entries : Backlog : Recent Change Rate
-----
server1.example.com:1389 : austin  : 10003   : 0       : 0
server2.example.com:1389 : austin  : 10003   : 0       : 0
```

A Deployment with Non-Interactive dsreplication

This example will create a four-server topology spanning two data centers. The four servers are already installed and have locations Austin and Budepest defined.

To Deploy with Non-Interactive dsreplication

1. Generate the sample data using the `make-ldif` tool on the first server.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
--ldifFile ldif/10K.ldif
```

2. Select a server from which to import data, and to be the source for future initialization to other servers. Stop this server, import the sample LDIF and start again, or perform a task-based `import-ldif` with the connection options.

```
$ bin/stop-ds
$ bin/import-ldif --backendID userRoot --ldifFile ldif/10K.ldif
$ bin/start-ds
```

3. Enable replication by choosing a specific server and running `dsreplication enable` three times. For the first invocation, create the replication topology administrator with the name `admin`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
--bindDN1 "cn=Directory Manager" --bindPassword1 password \
--replicationPort1 8989 --host2 austin02.example.com --port2 1389 \
--bindDN2 "cn=Directory Manager" --bindPassword2 password \
--replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
--adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
--bindDN1 "cn=Directory Manager" --bindPassword1 password \
--replicationPort1 8989 --host2 budapest01.example.com --port2 1389 \
--bindDN2 "cn=Directory Manager" --bindPassword2 password \
--replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
--adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
--bindDN1 "cn=Directory Manager" --bindPassword1 password \
--replicationPort1 8989 --host2 budapest02.example.com --port2 1389 \
--bindDN2 "cn=Directory Manager" --bindPassword2 password \
--replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
--adminPassword password --no-prompt
```

4. Initialize the other servers (the `dc=example,dc=com` replicas) from the server that had the data imported with `import-ldif`. To minimize the WAN transfers, initialize `budapest02` from `budapest01`.

```
$ bin/dsreplication initialize --hostSource austin01.example.com --portSource 1389 \
--hostDestination austin02.example.com --portDestination 1389 \
--adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-prompt

$ bin/dsreplication initialize --hostSource austin01.example.com --portSource 1389 \
--hostDestination budapest01.example.com --portDestination 1389 \
--adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-prompt

$ bin/dsreplication initialize --hostSource budapest01.example.com --portSource 1389 \
--hostDestination budapest02.example.com --portDestination 1389 \
--adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-prompt
```

5. View the state of replication.

```
$ bin/dsreplication status --adminPassword password --no-prompt --displayServerTable
--showAll
```

To Use `dsreplication` with SASL GSSAPI (Kerberos)

This example procedure assumes that you have configured SASL GSSAPI on all servers in the replication topology and that they are working properly.

The Directory Server's utilities all support SASL GSSAPI options for systems using Kerberos as its main authentication mechanism. The following procedure shows how to use `dsreplication`

with SASL GSSAP to set up a new `replication.admin` identity, while enabling replication on a server. The following are important points about the configuration:

- A separate Kerberos identity is required to manage replication. Existing Kerberos credentials can be used to interact with the server when enabling replication and creating the new identity.
- The new identity, such as `replication.admin`, must not exist as the `cn` or `uid` value under any public base DN.

1. Set the LDAP Connection Handler to explicitly listen on the server's hostname address.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --remove listen-address:0.0.0.0 --add listen-address:host.example.com
```

2. Update the identity mapper to have `cn=admin data` included in the list of base DN's and to add the `cn` attribute to match attributes. This step is required to map the admin account to the Kerberos realm.

```
$ bin/dsconfig set-identity-mapper-prop \
  --mapper-name "Regular Expression" \
  --add match-attribute:cn \
  --set "match-base-dn:cn=admin data" \
  --set match-base-dn:dc=example,dc=com
```

3. Invoke `replication enable`, authenticating as the existing kerberos `authid`. Note that no bind DN's and passwords are required to authenticate because we are using SASL binding. However, the new replication admin user requires a password at creation time, so we recommend using a strong random password. Once SASL is working, you will no longer have to provide this random password. Also note that we are forcing the use of the ticket cache, so make sure you have properly authenticated as `ds.admin` from your local host and the ticket is not expired in the cache.

```
$ kinit -p ds.admin
$ bin/dsreplication enable \
  --host1 server1.example.com --port1 1389 --replicationPort1 1989 \
  --host2 server2.example.com --port2 2389 --replicationPort2 2989 \
  --baseDN dc=example,dc=com \
  --adminUID replication.admin --adminPassword strongPassword \
  --saslOption1 mech=gssapi --saslOption1 authid=ds.admin@EXAMPLE.COM \
  --saslOption1 useTicketCache=true --saslOption1 requireCache=true \
  --saslOption2 mech=gssapi --saslOption2 authid=ds.admin@EXAMPLE.COM \
  --saslOption2 useTicketCache=true --saslOption2 requireCache=true
```

4. Use `dsreplication initialize` to initialize data on remote server.

```
$ kinit -p replication.admin
$ bin/dsreplication initialize \
  --hostSource server1.example.com --portSource 1389 \
  --hostDestination server2.example.com --portDestination 2389 \
  --baseDN dc=example,dc=com \
  --saslOption mech=GSSAPI \
  --saslOption authID=replication.admin@EXAMPLE.COM \
  --no-prompt
```

5. The temporary `userPassword` can now be deleted from the `replication.admin` entry. Create a file called `remove-password.ldif` with these contents:

```
dn: cn=replication.admin,cn=Administrators,cn=admin data
changetype: modify
delete: userPassword
```

6. Apply the modifications using `ldapmodify`:

```
$ ./ldapmodify --filename remove-password.ldif -o mech=GSSAPI
-o authid=replication.admin@example.com \
--saslOption useTicketCache=true \
--hostname host.example.com --port 1389 \
--noPropertiesFile
```

7. Check the topology's status by running `dsreplication status`. The `--saslOption useTicketCache=true` and `--saslOption requireCache=true` properties, instead of providing a password, for all `dsreplication` commands after properly creating the admin accounts and mappers.

```
$ bin/dsreplication status \
--saslOption mech=gssapi \
--saslOption authid=replication.admin@EXAMPLE.COM \
--saslOption useTicketCache=true --saslOption requireCache=true \
--hostname host.example.com --port 1389 \
--no-prompt
```

Configuring Assured Replication

The Ping Identity Directory Server's replication mechanism is based on the eventual-consistency model, which is a loosely-connected topology that propagates updates to other servers without waiting for their corresponding replication response messages. As a result, there is a small window of time where updates are not all present on the replicas as the changes are replicating through the system. There are, however, deployments that require *immediate* access to replicated data. In such cases, administrators can configure *Assured Replication*, which ensures that replication has completed *before* the update response is returned to the client.

From the LDAP client's perspective, assured replication has no bearing on the result code of the operation, just on the time in which it takes to receive the response for those requests in which replication assurance is applied. Detailed information regarding assurance processing is available to an LDAP client with awareness of the assured replication control.

The assured replication mechanism takes advantage of server location to distinguish between local and remote servers to allow different policies to apply. For example, a common assurance approach is to respond to a client update after all servers in the same location have applied the update, and one or more servers in remote locations have received the update. In addition, the level of assurance applied to each operation can be explicitly requested by the client and/or specified by the server configuration using the Replication Assurance Policy.

Assured replication is supported by client requests directly to Ping Identity Directory Server and/or through an Ping Identity Directory Proxy Server.

About the Replication Assurance Policy

Assured replication uses a *Replication Assurance Policy* to define the properties needed to ensure that replication has satisfactorily completed before the update response is returned to the client. Multiple policies can be defined but only one policy is matched with a client update

request. Each policy contains an evaluation order index which, together with an optional request and connection criteria, provides flexibility in matching a policy to request.

The Replication Assurance Policy defines local and remote assurance *levels*. A *level* defines how rigorous the policy should be when waiting for propagation of updates, while *local* and *remote* distinguish servers in the same location versus servers in remote locations. Although optional, it is recommended that request or connection criteria be associated with a policy to apply replication assurance appropriately.

The Directory Server contains a Default Replication Assurance Policy that is enabled but has no assurance levels assigned. In addition to using the Default Replication Assurance Policy, any number of policies can be created and enabled. When a client request is received, the server iterates through the list of enabled policies according to each policy's *evaluation-order-index* property. A smaller *evaluation-order-index* value (e.g., 1) has precedence over policies with larger *evaluation-order-index* values (e.g., 2, 3, 4, etc.). The *evaluation-order-index* values do not need to be contiguous. The first policy that matches a request is associated with the operation.

The Replication Assurance Policy, which is defined on the Ping Identity Directory Server and not on the Ping Identity Directory Proxy Server, has the following properties:

- **evaluation-order-index.** Determines the evaluation order (the smaller value has precedence) among multiple Replication Assurance Policies that match against an operation.
- **local-level.** Specifies the assurance level used to replicate to local servers. A local server is defined as a server with the same `location` property value as set in the global configuration. The `local-level` property must be set to an assurance level as strict as the `remote-level` property. For example, if the `remote-level` is set to "processed-all-remote-locations," then the `local-level` property must be "processed-all-servers."
 - **None.** Replication to any local server is not assured.
 - **received-any-server.** At least one available local server must receive a replication update before a response is sent to the client.
 - **processed-all-servers.** All available local servers must complete replay of the replication update before the response is sent to the client. If a singular server is enabled, or only one server is available for a particular location, `processed-all-servers` will return a value of `false`.
- **remote-level.** Specifies the assurance level used to replicate to remote servers. A remote server is defined as a server that has a different `location` property value as set in the global configuration.
 - **None.** Replication to any remote server is not assured.
 - **received-any-remote-location.** At least one server at any available remote location must receive a replication update before a response is sent to the client.
 - **received-all-remote-locations.** All available remote servers must receive a replication update before the response is sent to the client.
 - **processed-all-remote-servers.** All available servers at all locations must complete replay of the replication update before the response is sent to the client. If a single server is enabled, or only one server is available for a particular location, `processed-all-remote-servers` will return a value of `false`.

- **timeout.** Specifies the maximum length of time to wait for the replication assurance requirements to be met before timeout and replying to the client.
- **connection-criteria.** Specifies connection criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and request criteria are specified for a policy, then both must match an operation for the policy to be assigned.
- **request-criteria.** Specifies request criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and a request criteria are specified for a policy, then both must match an operation for the policy to be assigned.

Servers in a replication topology are not required to share a homogeneous set of policies; you can configure the Replication Assurance Policies differently on the replicas in a topology. For example, if you configure server A to match add operations to a `processed-all-servers` assurance level, and server B to match add operations to a local `received-any-server` level, then add operations received on server A will have the assurance level of `processed-all-servers` and add operations received on server B will have the assurance level of `received-any-server`.

For more detailed information, see the *Ping Identity Directory Server Configuration Reference Guide*.

Points about Assured Replication

The following are some points when implementing Assured Replication:

- **Client Controls.** The client may optionally include an assured replication request control with each operation. This control allows the client to specify bounds on assurance levels and/or override the timeout assigned by the matching replication assurance policy for the associated operation. The server always honors these request controls. See [About the Assured Replication Controls](#) for more information.
- **Directory Proxy Server.** Replication assurance policies are not supported on the Ping Identity Directory Proxy Server. Replication client controls are passed through to the underlying Directory Server backend.
- **Schema and Admin Data Backend Replication.** The schema and admin data backends are not supported by Assured Replication. Replication assurance policies that include criteria to match either backend will be rejected.
- **Backward Compatibility.** Server versions that support assured replication are backwards-compatible with prior versions that do not support assured replication.
- **WAN-Friendly Replication.** Assured replication functions independently from WAN-Friendly Replication and the notion of WAN Gateways.
- **Global Configuration Properties.** The Directory Server provides two configurable global configuration properties that determine the timing of the assurance source and maximum number of replication backup updates to be recognized as an available source.
 - **replication-assurance-source-timeout-suspend-duration.** Specifies the amount of time a replication assurance source will be suspended from assurance requirements if it

experiences an assurance timeout. While suspended, the source will be excluded from assurance requirements for all operations originating on this Directory Server. This avoids the situation of repeated timeouts caused by degraded or offline servers. Default is 10 seconds.

- **replication-assurance-source-backlog-fast-start-threshold.** Specifies the maximum number of replication backlog updates a replication assurance source can have and be immediately recognized as an available source. If a source connects to this server with more than the configured threshold backlog updates, it will be excluded from assurance requirements for all operations originating from this Directory Server until it completes at least one assurance successfully (i.e. this Directory Server receives an update acknowledgement message from it within the timeout window). Default is 1000.

To Configure Assured Replication

The following example procedure assumes that you have set up three servers on localhost, on ports 1389, 2389 and 3389. Note that each server has a default Replication Assurance Policy with no assurance levels set.

1. On server 1, use `dsconfig` to create request criteria for add operations. This request criteria will be used to match any add operation with the Replication Assurance Policy that will be configured in the next step.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add
```

2. On server 1, set up the Replication Assurance Policy to make all add operations assured with a level of `processed-all-servers`, which indicates that all local servers in the topology must complete replay of the replication update before the response is sent to the client. Specify the Adds request criteria configured in the previous step.

```
$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Processed All Locally" \
  --set evaluation-order-index:1 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Adds
```

3. On server 1, repeat the previous two steps for modify operations. The Replication Assurance Policy "Mods Received Any Locally" ensures that at least one available local server must receive a replication modify update before a response is sent to the client.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Received Any Locally" \
  --set evaluation-order-index:2 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

- On server 2, repeat steps 1-3 to set up the Adds and Mods request criteria and its respective Replication Assurance Policy.

```
$ bin/dsconfig create-request-criteria \
--criteria-name Adds \
--type simple \
--set operation-type:add

$ bin/dsconfig create-request-criteria \
--criteria-name Mods \
--type simple \
--set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
--policy-name "Adds Received Any Locally" \
--set evaluation-order-index:1 \
--set local-level:received-any-server \
--set "timeout:500ms" \
--set request-criteria:Adds

$ bin/dsconfig create-replication-assurance-policy \
--policy-name "Mods Processed All Locally" \
--set evaluation-order-index:2 \
--set local-level:processed-all-servers \
--set "timeout:500ms" \
--set request-criteria:Mods
```

- Leave server 3 as is with the default Replication Assurance Policy configured with no assurance levels or criteria.
- On server 1, list the policies on the server using the dsconfig command to confirm that they exist on the server.

```
$ bin/dsconfig list-replication-assurance-policies
```

Replication Assurance Policy	Type	enabled	evaluation-order-index	local-level	remote-level
Adds Processed All Locally	generic	true	1	processed-all-servers	none
Mods Received Any Locally	generic	true	2	received-any-server	none
Default Replication Assurance Policy	generic	true	9999	none	none

- Repeat the previous step on server 2 and server 3. Server 3 should only show the Default Replication Assurance Policy.
- Check the Replication Assurance counters on all servers before any add or modify operation using ldapsearch. They should be set to zero. These counters are on the replica server, which is where the policy is matched and assigned. On server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
"(objectclass=*)" | grep replication-assurance

replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

- Check the Replication Summary table on all of the servers. For example, on server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replication Summary dc_example_dc_com,cn=monitor" \
"(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 0
```



```

replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0

```

- 10.** Add an entry to the server 1 Directory Server. Check that the counters matched the newly added entry to the "Adds Processed All Locally" Policy and that it completed assured.

```

$ bin/ldapmodify --filename add-user.ldif --defaultAdd

$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy: 0
replication-assurance-policy-matches: Mods Received Any Locally: 0
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-remote-level-uses: none: 1

$ bin/ldapsearch --baseDN "cn=Replication Summary dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0

```

- 11.** Perform a modify of an entry under dc=example,dc=com on server 1. Check that the counters matched the modify operation to the "Mods Processed All Locally" policy and that the operations completed assured.

```

$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy: 0
replication-assurance-policy-matches: Mods Received Any Locally: 1
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-local-level-uses: received-any-server: 1

```

```
replication-assurance-remote-level-uses: none: 2

$ bin/ldapsearch --baseDN "cn=Replication Summary dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

You have successfully configured Assured Replication.

About the Assured Replication Controls

The LDAP SDK for Java provides an implementation of an LDAP control that can be included in add, bind, modify, modify DN, and certain extended requests to indicate the level of replication assurance desired for the associated operation. The OID for this control is 1.3.6.1.4.1.30221.2.5.28, and may have a criticality of either TRUE or FALSE.

For specific details, see the LDAP SDK javadoc for the `AssuredReplicationRequestControl` class.

Managing the Topology

The following sections describe common topology management operations.



Note: When enabling or disabling replication within a topology that contains multiple product versions, the `dsreplication` tool must be run from the server root location of a member of the topology that has the oldest product version.

To Add a Server to the Topology

The following steps assume an existing directory server topology. The commands are identical for initial enable between two servers, where one server contains data for the replication domain stored in the `userRoot` backend. If database encryption is being used on the servers in the topology, it is important that the server being initialized has a copy of the `encryption-settings` backend from the source server.

1. All servers, in the topology and the new server, should be online.
2. Enable replication for the base DN, or base DN's, using an existing server as `host1` and the new server as `host2`.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
  --bindDN1 "cn=Directory Manager" --bindPassword1 password \
```

```
--replicationPort1 8989 --host2 austin03.example.com --port2 1389 \  
--bindDN2 "cn=Directory Manager" --bindPassword2 password \  
--replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \  
--adminPassword password --no-prompt
```

- Optionally, compare the configurations between the two hosts used in the `dsreplication enable` command. Make sure settings are consistent across the topology and are also consistent with the new system:

```
$ bin/config-diff --sourceLocal --targetHost austin03.example.com  
--targetBindDN "cn=directory manager" --targetBindPassword pass --targetPort 1389
```

Disabling Replication and Removing a Server from the Topology

When removing a server from the topology, the remaining servers need to be made aware of the change. If the server to be removed is online, then one invocation of `dsreplication disable` is all that is necessary. If the server to be removed is offline, then two steps are required: `dsreplication remove-defunct-server` from another server in the topology, and `dsreplication cleanup-local-server` on the offline server to be removed. Similar to the `enable` command, all servers not being removed from the topology need to be online during the process. The following examples show the steps in more detail:

- Removing a server that is still online.** The `dsreplication disable` command can be run from any server, but all servers in the topology need to be online.

```
$ bin/dsreplication disable --hostname austin03.example.com --port 1389 \  
--baseDN dc=example,dc=com --adminUID admin --adminPassword password \  
--no-prompt
```

- Removing a server that is offline.** The `dsreplication remove-defunct-server` command can be run against any server not being removed from the topology. All servers in the topology, except for the offline servers being removed, must be online.

```
$ bin/dsreplication remove-defunct-server --hostname austin01.example.com --port 1389 \  
--baseDN dc=example,dc=com --adminUID admin --adminPassword password \  
--no-prompt --defunctHost austin03.example.com --defunctPort 1389  
--defunctReplPort 8989
```

Run the `dsreplication cleanup-local-server` subcommand on each server removed from the topology to remove any topology references.

```
$ bin/dsreplication cleanup-local-server --no-prompt
```

Replacing the Data for a Replicating Domain

In the rare event that the data for the entire replication domain (such as the backend) needs to be replaced, perform the following steps:

To Replace the Data

1. With all servers online, the `dsreplication pre-external-initialization` command must be run once against any server in the topology. This stops replication for the domain. No writes are made by clients to any of the servers.

```
$ bin/dsreplication pre-external-initialization --hostname austin01.example.com \  
--port 1389 --baseDN dc=example,dc=com --adminUID admin \  
--adminPassword password --no-prompt
```

2. Using `import-ldif`, replace the data for `dc=example,dc=com`. Make sure that the input LDIF is free of any replication attributes by using the `--excludeReplication` option. The `--overwriteExistingEntries` option is necessary to overwrite the existing data for the domain. For example, to perform the `import-ldif` with the server offline:

```
$ bin/import-ldif --ldifFile new-data.ldif --backendID userRoot --excludeReplication \  
--overwriteExistingEntries
```

3. Initialize the other servers in the topology with `dsreplication initialize`, using the server which has the new data as the source host. For example:

```
$ bin/dsreplication initialize --hostSource austin01.example.com --portSource 1389 \  
--hostDestination budapest01.example.com --portDestination 1389 \  
--adminUID admin --adminPassword password --baseDN dc=example,dc=com \  
--no-prompt
```

4. Run `dsreplication post-external-initialization` once from any server in the topology. All servers in the topology must be online:

```
$ bin/dsreplication post-external-initialization --hostname austin01.example.com \  
--port 1389 --baseDN dc=example,dc=com --adminUID admin \  
--adminPassword password --no-prompt
```

Advanced Configuration

The following sections are advanced configuration procedures that may be appropriate for your company's deployment.

Changing the replicationChanges DB Location

You can change the `replicationChanges` DB location if on-disk space issues arise. The replication changelog database can live outside `<server-root>` and be placed in another location on the filesystem. In that case, you must specify the absolute path of the replication changelog directory.

To Change the replicationChanges DB Location

1. Use `dsconfig` to change the database location for the replication changelog, which by default is at `<server-root>/changelogDb`. The following command sets the replication changelog backend to `<server-root>/data/directory/changelogDB`. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-db-directory:/data/directory/changelogDb" \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

2. Stop the server, move the DB files, and then restart the server.

```
$ bin/stop-ds
$ mv changelogDb /data/directory
$ bin/start-ds
```

Modifying the Replication Purge Delay

The replication purge delay specifies the period after which the directory server purges changes on the replication server database. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

Currently, the Ping Identity Directory Server sets the default purge delay to one day. Administrators can change the default purge delay using the `dsconfig` tool. To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

To Modify the Replication Purge Delay

- Use `dsconfig` to change the purge delay. The property accepts time units of seconds (s), minutes (m), hours (h), days (d), or weeks (w). The following command is entered on the command line and changes the purge delay from the default one day to two days.

In `dsconfig` interactive mode, open the **Advanced objects** menu. Select **Replication Server**. Select the replication synchronization provider, and then select the option to change the replication purge delay.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-purge-delay:2 d"
```

Configuring a Single Listener-Address for the Replication Server

By default, the replication server binds the listening ports to all available interfaces of the machine. To bind the listener to a specific address, the address must be the hostname provided when replication is enabled and the `listen-on-all-addresses` property must be set to `FALSE`.

The replication server's configuration entry already stores a host name for itself so that it can resolve the address and specify it during the socket bind. If the server information is missing from the system, an error message will be generated with instructions on specific address binding. You can use the `dsconfig` tool to change the value of the `listen-on-all-addresses` property from `TRUE` (default) to `FALSE`.

To Configure a Replication Server to Listen on a Single Address

1. Create a new directory server instance with replication enabled on port 8989.
2. Run `netstat` to see the ports bound for listening on port 8989. Notice that `*.8989` means that it is listening on all addresses.

```
$ netstat -an | grep LISTEN | grep 8989
```

3. Run `dsconfig` to disable listening on all addresses for the replication server.

```
$ bin/dsconfig set-replication-server-prop \  
  --provider-name "Multimaster Synchronization" \  
  --set listen-on-all-addresses:false
```

4. Run `netstat` again to see the ports bound for listening on port 8989. Notice that `<address>.8989` (for example, `10.8.1.211.8989`) means that it is listening on the one address.

Monitoring Replication

Replication in the Ping Identity Directory Server can be monitored the following ways:

- The `dsreplication status` subcommand displays basic information about the replicated base DNs, the number of entries replicated as well as the approximate size of replication backlogs at each Directory Server.
- The more detailed information about the state of replication can be obtained via the information exposed in its monitoring framework under `cn=monitor`. Administrators can monitor their replication topologies using several tools and protocols: the Data Metrics Server, SNMP, LDAP, JMX, or through the Administrative Console. See [Managing Logging and Monitoring](#).

- The Periodic Stats Logger plug-in allows collecting replication statistics for profiling server performance. For more information, see [Profiling Server Performance Using the Periodic Stats Logger](#).

Monitoring Replication Using cn=monitor

The `cn=monitor` branch has a number of entries that store the replication state of a topology.

- **Direct LDAP Server** `<baseDN> <host name:port> <serverID>`. Defines an LDAP server that is directly connected to the replication server that you are querying. The information in this entry applies to the replication server local to the `cn=monitor` entry. For detailed information, see [Summary of the Direct LDAP Monitor Information](#).
- **Indirect LDAP Server** `<baseDN> <serverID>`. Defines an LDAP server that is connected to another replication server in the topology. While this server is connected to the same topology, it is not connected to the replication server being queried. For detailed information, see [Summary of the Indirect LDAP Server Monitor Information](#).
- **Remote Repl Server** `<baseDN> <host name:port> <serverID>`. Defines a remote replication server that is connected to the local replication server. Information in this entry is in respect to the Replication Server local to the `cn=monitor` branch. For detailed information, see [Summary of the Remote Replication Server Monitor Information](#).
- **Replica** `<baseDN>`. Stores information for an instance of the replicated naming context—also known as the replica—with respect to the Directory Server and its communication with a replication server. The Replica information is what is responsible for sending and receiving changes from the replication servers. For detailed information, see [Summary of the Replica Monitor Information](#).
- **Replication Server** `<replPort> <serverID>`. Shows the information specific to the Replication Server running, for example, on the replication port `<replPort>` with a server ID of `<serverID>`. This entry defines the replication server. For detailed information, see [Summary of the Replication Server Monitor Information](#).
- **Replication Server Database** `<baseDN> <serverID>`. Shows information for the changelog table of replica (suffix) in the replication server. As the Replication Server receives updates from the Directory Server, it records those changes in the changelog table. For detailed information, see [Summary of the Replication Server Database Monitor Information](#).
- **Replication Server Database Environment** `<baseDN>`. Shows the information for the database environment for the replication server backend plus the total number of records added to and deleted from the database. For detailed information, see [Summary of the Replication Server Database Environment Monitor Information](#).
- **Replication Summary** `<baseDN>`. Shows summary information on the replication topology and the state for a particular base DN. For detailed information, see [Summary of the Replication Summary Monitor Information](#).
- **Replication Changes Backend**. Shows the backend information for all replication changes. For detailed information, see [Summary of the Replication Changes Backend Monitor Information](#).

- **Replication Protocol Buffer.** Shows the state of the buffer (initially 4k) for protocol operations, which is stored in thread local storage. For detailed information, see [Summary of the Replication Protocol Buffer Monitor Information](#).

Replication Best Practices

The following are recommended best practices related to replication based on our observations in actual production environments.

About the dsreplication Command-Line Utility

The following points involve some security practices as applies to replication. For specific questions, please contact your authorized support provider.

- **Developing Scripts.** The `dsreplication` utility maintains the history of executed `dsreplication` commands with the full command-line arguments in the `logs/tools/dsreplication.history` file. The recorded commands may be used to develop scripts to set up and configure replication.

Scripts invoking the `dsreplication` utility in non-interactive mode should check the return code from the `dsreplication` process. A non-zero return code indicates some sort of failure.

If output messages from the `dsreplication` utility are not desired, use the `--quiet` option to suppress them.

The utility, by default, fails if one or more warnings are issued during the command execution. Warnings can be suppressed using the `--ignoreWarnings` option. For example, this option is required when using `dsreplication` with non-fully-qualified hostnames (for example, `localhost`), otherwise `dsreplication` will fail. In production environments, use of this flag is strongly discouraged.

The `dsreplication` utility also provides an `--ignoreLock` option that specifies that the tool should continue processing in non-interactive mode or in scripts even if the replication topology has been locked by another invocation of `dsreplication`. However, this option should be used with caution.

- **Concurrent Use.** With the exception of the `dsreplication status` subcommand, the `dsreplication` subcommands cannot be executed concurrently. The command-line utility locks the replication topology at one or more servers to prevent accidental configuration changes caused by multiple `dsreplication` subcommands running at the same time. It is best to avoid concurrent configuration changes in general.
- **Status.** The `dsreplication status` subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of `dsreplication status` is displayed. By default, `dsreplication` will display the status for all replicated domains (with the exception of the special domains of the schema and the server registry).

It is recommended to select a particular base DN for `dsreplication status` if multiple base DNs are configured for replication.

It is also recommended to avoid invoking `dsreplication status` too often (more than once every 15 seconds) or from multiple locations at the same time. Some of the information displayed by `dsreplication status` is based on monitor information that is not refreshed every time the monitor is queried.

The `status` subcommand should not be used for collecting performance metrics. It is best to rely on replication-related information captured by the Periodic Stats Logger Plug-in.

Replication Conflicts

This section provides more in-depth information on replication conflicts than presented in earlier sections, so that administrators can understand the mechanisms and possible scenarios behind these conflicts.

Updates to Directory Server entries in a replication topology may happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model also means that conflicting changes can be applied at different directory server instances. In most cases, the Directory Server is able to resolve these conflicts automatically and in a consistent manner (i.e., all directory server instances in a replication topology will resolve each and every conflict the same way). However, in some scenarios, as seen below, manual administrative action is required. For any of these unresolved conflicts, the administrator is notified via administrative alerts.

On a high-level, the conflict resolution algorithm tries to resolve conflicts as if the operations causing the conflict in a distributed environment has been applied to a single directory server instance. For example, if the same entry is added to two different directory server instances at about the same time, then once these operations have been replicated, both directory servers will keep only the entry that was added first. The following figure highlights the differences between standalone versus replicated environments.

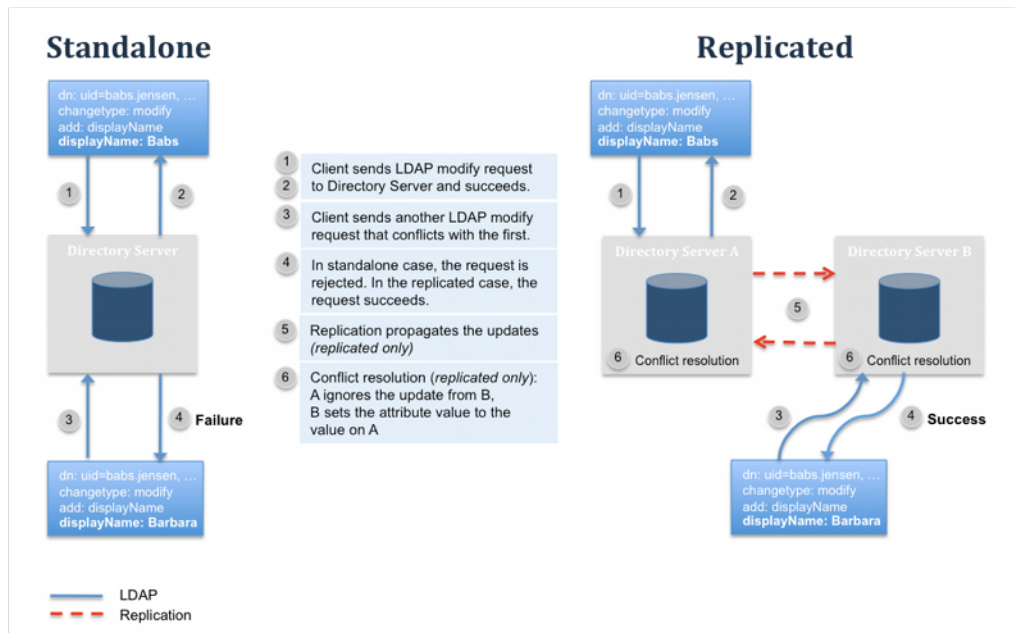


Figure 13: Conflicting Operations in Standalone versus Replicated Environments

Types of Replication Conflicts

There are fundamentally two types of replication conflicts: naming and modification conflicts. Naming conflicts include operations that cause conflicts with the naming (DN) of the existing or new entries, while modification conflicts include operations that result in conflicts in the modification of attributes.

Naming Conflict Scenarios

For all of the naming conflict scenarios in the table below, assume the following:

- Update 1 was applied at Directory Server 1
- Update 2 was applied at Directory Server 2
- Update 1 occurred shortly before Update 2, so that Directory Server 2 received Update 1 after Update 2 was applied

The naming conflict scenario is illustrated in the following figure:

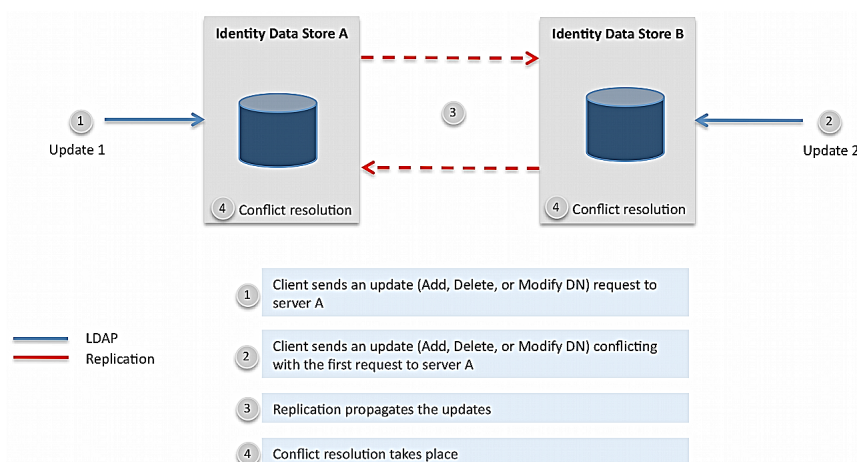


Figure 14: Naming Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This may lead to inconsistent replays.

Table 45: Naming Conflict Scenarios

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify	Delete	Yes	Modify is discarded.
Modify	Modify DN	Yes	New entry is located based on the entryUUID and Modify is applied to the renamed entry.
Delete	Delete	Yes	Delete operation is ignored.
Delete	Modify DN	Yes	Delete operation is applied to the renamed entry.
Delete of A	Add of B under A	Yes	Entry B is renamed and Entry A is deleted.
Modify DN	Delete of the same entry targeted by the Modify DN	Yes	Modify DN operation is ignored.
Modify DN with a new parent	Delete of parent	No	The entry targeted in the Modify DN operation is marked as a conflict entry.
Modify DN with a new parent	Modify DN of the parent	Yes	The entry will be moved under the new DN of the parent.
Modify DN of A with new DN B	Modify DN of C with new DN B	No	A and B will be conflict entries.
Add A	Modify DN of the parent of A	Yes	The entry is added under the new DN of the parent.
Add A	Delete of the parent of A	No	The added entry is marked as a conflict entry.
Add A	Add A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Add A	Add A with different set of attributes (or values)	No	The existing entry is marked as a conflicting entry and the incoming Add is executed.

Modification Conflict Scenarios

Modification conflicts are always resolved automatically and no manual action is required. The LDAP Modify operation allows the following modification types:

- > Add of one or more values
- > Delete of one or more values or the entire attribute
- > Replacement of all values

Replication does not currently support the increment LDAP modification type.

For all of the operations in the table below, assume the following:

- > LDAP Modify 1 was applied at Directory Server 1
- > LDAP Modify 2 was applied at Directory Server 2
- > LDAP Modify 1 occurred shortly before LDAP Modify 2, so that Directory Server 2 received LDAP Modify 1 after LDAP Modify 2 was applied.

The modification conflict scenario is illustrated in the figure below:

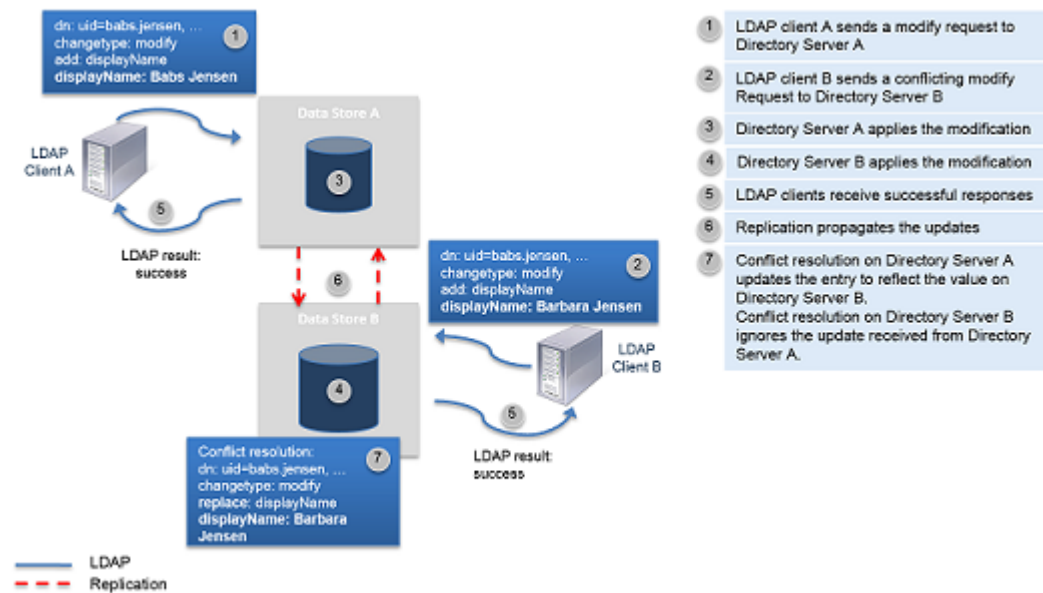


Figure 15: Modification Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs:

Table 46: Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Add of a single-value attribute	Add of the same attribute with a different value	Incoming Modify is ignored.
Delete of a single-valued attribute	Replacement of the value of the same attribute	Incoming Delete is ignored.
Replacement of a single-valued attribute	Delete of the same attribute	Incoming Replacement is ignored.
Delete some values from a multi-valued attribute	Delete some values from a multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Add the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Delete value X from the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Add value Y to the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Delete value Y from the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Replace all values of the same multi-valued attribute	Incoming Delete is ignored.
Add of values X and Y to a multi-valued attribute	Delete value X from the same multi-valued attribute	Only value Y is added.
Delete value X from a multi-valued attribute	Add of values X and Y to the same multi-valued attribute	Incoming Delete is ignored.

Troubleshooting Replication

The following sections provide information to troubleshoot your replication deployment.

Recovering a Replica with Missed Changes

If a server has been offline for a period of time longer than the replication purge delay, the `dsreplication initialize` command must be performed to bring the replica into sync with the topology.

Server startup is the only time missed changes are detected. A missed change is a change that the replica detects that it needs, but which cannot be found within any other replication server's replicationChanges backend (stored in the path `server root /changelogDb`). If missed changes are detected, the server enters lockdown mode, where only privileged clients can make requests. Any other server that is not missing changes can be used as a source for `dsreplication initialize`.

If a manual backup and restore of the server is required, then the following steps are equivalent to `dsreplication initialize`.

Performing a Manual Initialization

In the event that an online initialization is not possible, the following steps can be used to initialize a server.

1. From another server in the replication topology, backup the `userRoot`, `adminRoot`, `schema`, `replicationChanges` backends to the `/bak` directory. If encrypted attributes are present, then the `encryption-settings` backend should also be exported. One or more encryption settings IDs may need to be exported and imported.

```
$ <source-server-root>/bin/backup --backendID userRoot -d bak/userRoot
$ <source-server-root>/bin/backup --backendID adminRoot -d bak/adminRoot
$ <source-server-root>/bin/backup --backendID schema -d bak/schema
$ <source-server-root>/bin/backup --backendID replicationChanges -d bak/
replicationChanges
$ <source-server-root>/bin/encryption-settings export --id ID --output-file bak/
exported-key
```

2. Copy the `bak` directory to the new replica.

```
$ scp -r <source-server-root>/bak <user>@<destination-server>:<destination-server-
root>/bak
```

3. Stop the server and restore the `userRoot`, `changelog`, `adminRoot`, `replicationChanges` backends. If the `encryption-settings` backend was exported, it should also be reimported.

```
$ <destination-server-root>/bin/restore -d bak/userRoot
$ <destination-server-root>/bin/restore -d bak/adminRoot
$ <destination-server-root>/bin/restore -d bak/schema
$ <destination-server-root>/bin/restore -d bak/replicationChanges
$ <destination-server-root>/bin/encryption-settings import --input-file bak/exported-
key --set-preferred
Enter the PIN used to encrypt the definition:
```

4. Start the server using `bin/start-ds`.

Fixing Replication Conflicts

Replication conflicts can occur when an incompatible change to an entry is made on two replicas at the same time. The change is processed on one replica and then replicated to the other replica, which causes the conflict. While most conflicts are resolved automatically, some require manual action.

To fix replication conflicts, initialize the replica containing the conflicts with the data from another replica that does not have conflicts. If the database is large and the number of conflicts small, running `ldapmodify` against the server with the conflict will work if the command includes the Replication Repair Control specified by OID value 1.3.6.1.4.1.30221.1.5.2. The Replication Repair Control prevents the change from replicating. It also enables changing operational attribute values, which are not normally writable.

The following steps provide an example of using the Replication Repair Control to fix replication conflicts by applying change to only the server with the conflict. There are two examples: one for a modification conflict found by performing an `ldap-diff`, and the other for a naming conflict.

To Fix a Modify Conflict

1. The `bin/ldap-diff` tool can be used to isolate conflicting entries between two replicas. The following uses the tool to search across the entire base DN for any difference in user attributes, and reports the difference in `difference.ldif`. Replace the `sourceHost` value with the server that needs the adjustment.

```
$ bin/ldap-diff --sourceHost austin02.example.com --sourcePort 1389 \
--sourceBindDN "cn=Directory Manager" --sourceBindPassword
pass \
--targetHost austin01.example.com --targetPort 1389 \
--targetBindDN "cn=Directory Manager" --targetBindPassword
--baseDN "dc=example,dc=com" --outputLDIF difference.ldif \
--searchFilter "(objectclass=*)" --numPasses 3 "*" pass \
"^userPassword"
```

2. The `difference.ldif` file is in a format that can be used with `ldapmodify` to apply changes to the server that contains conflicts. The `ldap-diff` command must have been run with the `sourceHost` value as the server with conflicts. The following is an example of the contents of `difference.ldif`:

```
dn: uid=user.1,ou=people,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 568 232 6789
-
delete: mobile
mobile: +1 568 591 7372
-
```

3. Run `bin/ldapmodify` to correct the entries on only the server with conflicts.

```
$ bin/ldapmodify --bindPassword password -J "1.3.6.1.4.1.30221.1.5.2" \
--filename difference.ldif
```

To Fix a Naming Conflict

1. In this example, a naming conflict was encountered when the replica attempted to replay an ADD of `uid=user.200,ou=people,dc=example,dc=com`. In other words, two clients added the entry at the same time as an entry of the same name was added on another replica.

```
[18/Feb/2010:14:53:12 -0600] category=EXTENSIONS severity=SEVERE_ERROR
msgID=1880359005 msg="Administrative alert type=replication-unresolved-conflict
id=bbd2cbaf-90a4-42af-94a8-cla42df32fc6
class=com.unboundid.directory.server.replication.plugin.ReplicationDomain
msg='An unresolved conflict was detected for DN
uid=user.200,ou=People,dc=example,dc=com.
The conflicting entry has been renamed to
entryuuid=69807e3d-ab27-43a3-8759-
ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com'"
```

2. The Directory Server prepends the entryUUID to the DN of the conflicting attribute and adds a `ds-sync-conflict-entry` auxiliary object class to the entry to aid in search. For example, the following command searches for any entry that has the `ds-sync-conflict-entry` objectclass and returns only the DNs that match the filter. You should see the conflicting entry for `uid=user.200`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub \
"(objectclass=ds-sync-conflict-entry)" "1.1"

dn: entryuuid=69807e3d-ab27-43a3-8759-
ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com

dn: entryuuid=523c430e-
a870-4ebe-90f8-9cd811946420+uid=user.200,ou=People,dc=example,dc=com
```



Note: Conflict entries are not returned unless the `objectclass=ds-sync-conflict-entry` is present in the search filter.

- After comparing the conflict entry with the target entry, the difference can be applied in a manner similar to the previous example using `ldapmodify` with the Replication Repair Control. The conflict entry can also be deleted using this command. Run `bin/ldapmodify` with the Replication Repair Control to make the fix. When making changes using the Replication Repair Control, the updates will not be propagated via replication. You should examine each and every replica one by one, and apply the necessary modifications using the request control.

```
$ bin/ldapmodify -J "1.3.6.1.4.1.30221.1.5.2" \
--filename difference.ldif
```

Fixing Mismatched Generation IDs

A warning that multiple generation IDs were detected for a specific suffix indicates that one or more replicas need to be re-initialized. If the warning is presented from a server after an initialization, it could be that `post-external-initialization` was not run as part of a global change in data. Running this command will fix the situation. The `dsreplication status` command will warn when any generation IDs are different across the topology.

Replication Reference

The following section shows general reference information related to replication.

Summary of the dsreplication Subcommands

A summary of the `dsreplication` subcommands and functions is presented in the table below.

Table 47: dsreplication subcommands

Subcommand	Description
<code>cleanup-local-server</code>	Removes replication-related artifacts from the configuration, schema as well as the server registry while the local server is offline. The subcommand does not remove references to this server from other replicas and replication servers in the topology. Therefore, it is recommended to remove this server first from the replication topology either by using the <code>disable</code> or <code>remove-defunct-server</code> subcommands. Since this subcommand can only be executed when the server is offline, replication attributes

Subcommand	Description
	<p>from suffixes other than the server registry or the schema will not be removed. The tool will produce an LDIF file, <code>logs/cleanup-backends.ldif</code> that may be used to the remove replica state from the base entry of these suffixes after the server is restarted.</p> <p>To remove the replication history from regular suffixes, export the formerly replicated suffixes using the <code>--excludeReplication</code> option of the <code>export-ldif</code> command. The resulting LDIF file can be re-imported using the <code>import-ldif</code> command. For example:</p> <pre data-bbox="630 499 1437 562">\$ bin/export-ldif --backendID userRoot --excludeReplication \ --ldifFile cleansed.ldif \$ bin/import-ldif --backendID userRoot --ldifFile cleansed.ldif</pre> <p>Exporting using the <code>--excludeReplication</code> option of the <code>export-ldif</code> command will also remove the replica state from the output. The LDIF created by the <code>cleanup-local-server</code> subcommand does not need to be applied after the server is restarted.</p>
disable	Disables replication on the specified server for the provided base DN and removes references to this server in the other servers with which it is replicating data.
enable	Updates the configuration of the servers to replicate the data under the specified base DN. If one of the servers is already replicating the data under the base DN with other servers, executing this subcommand will update the configuration of all the servers (so it is sufficient to execute the command line once for each server you add to the replication topology).
initialize	Initializes the data under the specified base DN on the destination server with the contents on the source server.
initialize-all	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server.
post-external-initialization	Used with <code>pre-external-initialization</code> , the command resets the generation ID based on the newly-loaded data. This subcommand must be called after initializing the contents of all the replicated servers using the <code>import-ldif</code> tool or <code>dsreplication initialize</code> . Specify the list of base DN's that have been initialized and provide the credentials of any of the servers that are being replicated. See the usage of the <code>pre-external-initialization</code> subcommand for more information. This subcommand only needs to be run on one of the replicas once.
pre-external-initialization	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the replication changelog database at each and every replication server. This subcommand should be used when globally restoring the replicas on all of the servers in a topology. You must specify the list of base DN's that will be initialized and provide the credentials of any of the servers that are being replicated. After calling this subcommand, initialize the contents of all the servers in the topology, then call the <code>post-external-initialization</code> subcommand. This subcommand only needs to be run on one of the replicas once.
remove-defunct-server	Removes an offline defunct server from the replication on all servers in the topology.
status	Displays the status of replication domains. If no base DN's are specified as parameters, the information for all base DN's is displayed. Available options with the status subcommand are: <code>--showAll</code> , <code>--displayServerTable</code> , <code>--location</code> .

Summary of the Direct LDAP Monitor Information

The following table provides a description of the attributes in the `cn=Direct LDAP Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor
```

Table 48: Direct LDAP Monitor Information

Monitor Attribute	Description
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this server is connected. The first number is the replication server port number and the second number is the server-id of the replication server.
replica-id:<serverID>	Replica ID number.
replication-backlog	Number of changes that the replication server has not seen from the server.
missing-changes	Number of missing changes.
approximate-delay	Difference between the time of the last change that the replication server has seen from the LDAP server and the most current time stamp on the latest change on the server.
base-dn	Base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Displays the replication protocol version.
generation-id	Generation ID for the base DN on the Directory Server.
restricted	Boolean that indicates whether the replication domain is restricted in an Entry Balancing Configuration with the Directory Proxy Server.
ack-sent	Number of acknowledgement messages sent to this replica (not currently used).
ack-received	Number of acknowledgement messages received from this replica (not currently used).
add-sent	Number of protocol messages with an LDAP Add sent to this replica.
add-received	Number of protocol messages with an LDAP Add received from this replica.
delete-sent	Number of protocol messages with an LDAP Delete sent to this replica.
delete-received	Number of protocol messages with an LDAP Delete received from this replica.
done-sent	Number of done messages sent to this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has been initialized over the replication protocol.
done-received	Number of done messages received from this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has completed the initialization of other replicas over the replication protocol.
entry-sent	Number of entry messages sent to this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to this replica.
error-received	Number of error messages received from this replica.

Monitor Attribute	Description
heartbeat-sent	Number of heartbeat messages sent to this replica.
heartbeat-received	Number of heartbeat messages received from this replica (should always be 0, since the replica never sends a heartbeat message to the server).
initialize-request-sent	Number of initialize-request messages sent to this replica. This message is sent when another replica requested initialization of its data using the replication protocol.
initialize-request-received	Number of initialize-request messages received from this replica. This message is sent when this replica requested initialization of its data using the replication protocol from another replica.
initialize-target-sent	Number of initialize-target messages sent to this replica. This message is sent before another replica has started the initialization of this replica.
initialize-target-received	Number of initialize-target messages received from this replica. This message is sent before this replica starts the initialization of one or more replicas.
modify-sent	Number of protocol messages with an LDAP Modify sent to this replica.
modify-received	Number of protocol messages with an LDAP Modify received from this replica.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to this replica.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from this replica.
repl-server-start-sent	Number of replication-server-start messages sent to this replica (should never be more than 1). The Replication Server responds with this message to the start message received from the replica.
repl-server-start-received	Number of replication-server-start messages received from this replica (should always be 0).
reset-generation-id-sent	Number of reset generation ID messages received from this replica.
reset-generation-id-received	Number of reset generation ID messages sent to this replica (should always be 0).
server-start-sent	Number of server-start messages sent to this replica (should always be 0).
server-start-received	Number of server-start messages received from this replica (should never be more than 1). Server-start is the first message the replica sends after establishing a replication connection.
window-sent	Number of window messages sent to this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-probe-sent	Number of window probe messages sent to this replica (should always be 0).
window-probe-received	Number of window probe messages received from this replica. The replica sends a window probe message to the server if the send window in the replica is closed and the replica is unable to publish updates to the server.
update-sent	Number of changes sent to this server.
update-received	Number of changes received from this server.
internal-connection	Indicates if the replica is in the same process as the replication server.
server-state	Displays the state of the replica. Displays the latest change number that the replica has seen from all the other replicas including itself.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.

Monitor Attribute	Description
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the LDAP server before requiring an ACK.
current-send-window	Current number of changes remaining to be sent to the LDAP server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received by the LDAP server before sending an ACK.
current-rcv-window	Number of changes remaining to be received by the LDAP server before sending an ACK Server.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Indirect LDAP Server Monitor Information

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry. These attributes provide information about a Directory Server that is connected to a different replication server in the topology.

```
dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor
```

Table 49: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
replica-id: <serverID>	ID number identifying the replica.
base-dn: <baseDN>	Base DN
connected-to: Remote Repl Server <baseDN> <host name:replPort> <replID>	Replication server to which the directory server is connected.
replication-backlog	Number of changes that the replication server has not seen from the server.
approximate-delay	Amount of time between the last change seen by this Directory Server and the most recent change seen by the remote replication server. This value is the amount of time between the time stamps, not the amount of time required to synchronize the two servers.
generation-id	Generation ID for this suffix on this remote replication server.
consumed-update-recent-rate	Rate that the connected Replication Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.

Monitor Attribute	Description
consumed-update-peak-rate	Highest rate that the connected Replication Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.

Summary of the Remote Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor
```

Table 50: Remote Replication Server Monitor Information

Monitor Attribute	Description
replication-server: <host name>:<repl port>	Host name and replication port number of the Replication Server.
replication-server-id:<serverID>	Server ID for the Replication Server.
available	Indicates if the remote replication server is available or not. Values: true or false.
sending-paused	Indicates if sending is paused. Values: true or false.
receiving-paused	Indicates if receiving is paused. Values: true or false.
wan-gateway-priority	Specifies the WAN Gateway priority of the remote replication server.
is-wan-gateway	Indicates if the remote replication server is a WAN Gateway. Values: true or false.
wan-gateway-desired	<p>Indicates if the remote replication server is a desired gateway. Values: true or false. This entry together with the <code>is-wan-gateway</code> property indicates the desired state.</p> <ul style="list-style-type: none"> > is-wan-gateway=false, wan-gateway-desire=false: Indicates another server with a higher gateway priority exists or the gateway priority is set to disabled. > is-wan-gateway=false, wan-gateway-desire=true: Indicates that the remote replication server wants to be a WAN gateway. > is-wan-gateway=true, wan-gateway-desire=false: Indicates that the remote replication server wants to give up its role as a WAN gateway. > is-wan-gateway=false, wan-gateway-desire=true: Indicates the remote replication server wants to remain as a gateway server.
base-dn	base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Replication protocol version.
generation-id	Generation ID for this suffix on this remote replication server.
restricted	Indicates that remote replication server is in an entry-balancing deployment.
add-sent	Number of protocol messages with an LDAP Add sent to the remote replication server.
add-received	Number of protocol messages with an LDAP Add received from the remote replication server.

Monitor Attribute	Description
delete-sent	Number of protocol messages with an LDAP Delete sent to the remote replication server.
delete-received	Number of protocol messages with an LDAP Delete received from the remote replication server.
done-sent	Number of done messages sent to the remote replication server. A done message indicates an end of online initialization session.
done-received	Number of done messages received from the remote replication server. A done message indicates an end of online initialization session.
entry-sent	Number of entry messages sent to the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to the remote replication server.
error-received	Number of error messages received from the remote replication server.
heartbeat-sent	Number of heartbeat messages sent to the remote replication server.
heartbeat-received	Number of heartbeat messages received from the remote replication server.
initialize-request-sent	Number of initialize-request messages sent to the remote replication replication server. This message is used during online initialization from one replica to another.
initialize-request-received	Number of initialize-request messages received from the remote replication server. This message is used during online initialization from one replica to another.
initialize-target-sent	Number of initialize-target messages sent to the remote replication server. This message is used before online initialization from one replica to another.
initialize-target-received	Number of initialize-target messages received from the remote replication server. This message is used before online initialization from one replica to another.
modify-sent	Number of protocol messages with an LDAP Modify sent to the remote replication server.
modify-received	Number of protocol messages with an LDAP Modify received from the remote replication server.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to the remote replication server.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from the remote replication server.
monitor-sent	Number of monitor messages sent to the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-received	Number of monitor messages received from the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-request-sent	Number of monitor requests sent to the remote replication server. The receiving server will respond with a monitor message that includes server's information about the state of the topology.

Monitor Attribute	Description
monitor-request-received	Number of monitor requests received from the remote replication server. This server will respond with a monitor message that includes server's information about the state of the topology.
monitor-v2-sent	Number of monitor messages sent to the remote server. This monitor message is only used with Directory Server v3.5 or later.
monitor-v2-received	Number of monitor messages received from the remote server. This monitor message is only used with Directory Server v3.5 or later.
pause-sending-updates-sent	Number of pause-sending-updates messages sent to the remote server. The remote server must stop sending update messages to this server when receiving this message.
pause-sending-updates-received	Number of pause-sending-updates messages received from the remote server. This server will stop sending update messages to the remote server upon receiving this message.
repl-server-start-sent	Number of replication-server-start messages sent to the remote replication server (should never be more than 1). The Replication Server responds with this message to the replication-server-start message received from remote replication servers.
repl-server-start-received	Number of replication-server-start messages received from the remote replication server (should never be more than 1) window-sent: the number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
reset-generation-id-sent	Number of reset generation ID messages received from the remote replication server. This message is sent before and after the data is initialized in the topology.
reset-generation-id-received	Number of reset generation ID messages sent to the remote replication server. This message is sent before and after the data is initialized in the topology.
server-info-sent	Number of replication server information messages sent to the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
server-info-received	Number of replication server information messages received from the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
set-source-location-sent	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
set-source-location-received	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
start-sending-updates-sent	Number of start-sending-updates messages sent to the remote replication server. The remote server may only start sending updates to this server after receiving this message.
start-sending-updates-received	Number of start-sending-updates messages received from the remote server. Sending update messages to the remote server may only start after receiving this message.
window-sent	Number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.

Monitor Attribute	Description
window-received	Number of window messages received from the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
update-sent	Total number of updates sent.
update-received	Total number of updates received.
server-state	Displays the server state of the remote replication server. It displays the last change seen on the remote replication server.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the remote replication server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received from the remote replication server before sending an ACK.
current-rcv-window	Number of changes remaining to be received from the remote replication server before sending an ACK.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the remote replication server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Replica Monitor Information

The following table provides a description of the attributes in the `cn=Replica` monitor entry for a specific base DN.

```
dn: cn=Replica <baseDN>,cn=monitor
```

Table 51: Indirect LDAP Server Monitor Information

Monitor Attribute	Description
base-DN: <baseDN>	Specified base DN. The monitor entries track your company's base DN (or <code>dc=example,dc=com</code>), <code>cn=schema</code> , and <code>cn=admin</code> data.

Monitor Attribute	Description
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the <code>server-id</code> of the replication server.
lost-connections	Number of times the Directory Server has lost connection to a replication server.
received-updates	Number of updates that the Directory Server Replica has received from the connected replication server.
sent-updates	Number of updates sent to the replication server.
pending-updates	Number of updates pending to send to the replication server.
replayed-updates	Total number of updates from the replication server that have been replayed for this replica.
replayed-updates-ok	Number of updates for this replica that have been successfully replayed with no conflicts.
replayed-update-failed	Number of updates for this replica that were successfully replayed after automatically resolving a modify conflict.
resolved-modify-conflicts	Number of updates for this replica that were successfully resolved after a modify conflict.
resolved-naming-conflicts	Number of updates for this replica that were successfully resolved after a naming conflict.
unresolved-naming-conflicts	Number of updates for this replica that could not be replayed due to an unresolvable naming conflict.
replica-id	Server ID for this replica.
max-rcv-window	Maximum number of changes that the Directory Server Replica can receive at a time before sending an acknowledgment back to the replication server.
current-rcv-window	Current received window size for this replica.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-rcv-window	Number of changes remaining to be received from the replication server before it must send an ACK.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the replication server before requiring an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the Directory Server.
replication-backlog	Number of changes that are from this replica.

Summary of the Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

```
dn: cn=Replication Server <baseDN> <replServerID>,cn=monitor
```

Table 52: Replication Server Monitor Information

Monitor Attribute	Description
replication-server-id	Server ID for the Replication server ID.
replication-server-port	Port number on which the replication server listens for communication from other servers.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
Generation IDs by Base DN	List of generation IDs for each base DN on the server.
num-outgoing-replication-server-connections	Number of outgoing connections from the replication server.
num-incoming-replication-server-connections	Number of incoming connections into the replication server.
num-incoming-replica-connections	Number of incoming connections to the replica.

Summary of the Replication Server Database Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server database monitor` entry.

```
dn: cn=Replication Server database <baseDN> <replServerID>,cn=monitor
```

Table 53: Replication Server Database Monitor Information

Monitor Attribute	Description
database-replica-id: <replicaID>	Specifies the replication server ID.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
first-change	First change number that is in this replication database table for this suffix from this <code>server-id</code> . For example, an example entry looks like the following: 0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011
last-change	Last change number that is in this replication database table for this suffix from this <code>server-id</code> . For example, an example entry looks like the following: 0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011
queue-size	Number of changes in the replication server queue waiting to be sent to this remote replication server database.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
records-added	Displays the number of records changed or added to the DIT.
records-removed	Displays the number of records removed from the DIT.

Summary of the Replication Server Database Environment Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server Database Environment monitor` entry, which includes the environment variables associated with the Oracle Berkeley Database Java Edition backend.

```
dn: cn=Replication Server Database Environment, cn=monitor
```

Table 54: Replication Server Database Environment Monitor Information

Monitor Attribute	Description
je-version	Current version of the Oracle Berkeley Java Edition.
current-db-cache-size	Current DB cache size.
max-db-cache-size	Maximum DB cache size.
db-cache-percent-full	Percentage of the cache used by the Directory Server.
db-directory	Directory that holds the changelogDb file.
db-on-disk-size	Size of the DB on disk.
cleaner-backlog	Number of log files that must be cleaned for the cleaner to meet its target utilization.
random-read-count	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
random-write-count	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
sequential-read-count	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
sequential-write-count	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
nodes-evicted	Accumulated number of nodes evicted.
active-transaction-count	Number of currently active transactions.
num-checkpoints	Number of checkpoints. A checkpoint is a process that writes to your log files all the internal BTree nodes and structures modified as a part of write operations to your log files to facilitate a quick recovery.
checkpoint-in-progress: false	Indicates if a checkpoint is in progress.
total-checkpoint-duration-millis	Total time in milliseconds for all checkpoints.
average-checkpoint-duration-millis	Average time in milliseconds for all checkpoints.
last-checkpoint-duration-millis	Duration in milliseconds of the last checkpoint run.
last-checkpoint-start-time	Start time of the last checkpoint.
last-checkpoint-stop-time	Stop time of the last checkpoint.
millis-since-last-checkpoint	Time in milliseconds since the last checkpoint.
read-locks-held	Total read locks currently held.
write-locks-held	Total write locks currently held.
transactions-waiting-on-locks	Total transactions waiting for locks
je-env-stat-AdminBytes	Number of bytes of JE cache used for log cleaning metadata and other administrative structure.
je-env-stat-BufferBytes	Total memory currently consumed by log buffers, in bytes.
je-env-stat-CacheDataBytes	Total memory of cache used for data.
je-env-stat-CacheTotalBytes	Total amount of JE cache in use, in bytes.
je-env-stat-CleanerBacklog	Number of files to be cleaned to reach the target utilization.
je-env-stat-CursorsBins	Number of bottom internal nodes (BINs) encountered by the INCompressor that had cursors referring to them when the compressor ran. The compressor thread cleans up the internal BTree as records are deleted to ensure unused nodes are not present.
je-env-stat-DataBytes	Amount of JE cache used for holding data, keys and internal Btree nodes, in bytes.

Monitor Attribute	Description
je-env-stat-DbClosedBins	Number of bins encountered by the INCompressor that had their database closed between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-EndOfLog	Location of the next entry to be written to the log.
je-env-stat-InCompQueueSize	Number of entries in the INCompressor queue when the getStats() call was made.
je-env-stat-LastCheckpointEnd	Location in the log of the last checkpoint end.
je-env-stat-LastCheckpointId	ID of the last checkpoint.
je-env-stat-lastCheckpointStart	Location in the log of the last checkpoint start.
je-env-stat-lockBytes	Number of bytes of JE cache used for holding locks and transactions.
je-env-stat-NBINSStripped	Number of BINS stripped by the evictor.
je-env-stat-NCacheMiss	Total number of requests for database objects which were not in memory.
je-env-stat-NCheckpoints	Total number of checkpoints run so far.
je-env-stat-NCleanerDeletions	Number of cleaner file deletions this session.
je-env-stat-NCleanerEntriesRead	Accumulated number of log entries read by the cleaner.
je-env-stat-NCleanerRuns	Number of cleaner runs this session.
je-env-stat-NClusterLNsProcessed	Accumulated number of leaf nodes (LNs) processed because they qualify for clustering.
je-env-stat-NDeltaINFlush	Accumulated number of delta internal nodes (INs) flushed to the log.
je-env-stat-NEvictPasses	Number of passes made to the evictor.
je-env-stat-NFSyncRequests	Number of fsyncs requested through the group commit manager. <code>Fsync()</code> synchronizes the filesystem after a write or a transaction.
je-env-stat-NFSyncTimeouts	Number of fsync requests submitted to the group commit manager which timed out.
je-env-stat-NFSyncs	Number of fsyncs issued through the group commit manager.
je-env-stat-NFileOpens	Number of times a log file has been opened.
je-env-stat-NFullBINFlush	Accumulated number of full bottom internal nodes (BINS) flushed to the log.
je-env-stat-NFullINFlush	Accumulated number of full INs flushed to the log.
je-env-stat-NINsCleaned	Accumulated number of INs cleaned.
je-env-stat-NINsDead	Accumulated number of INs that were not found in the tree anymore (deleted).
je-env-stat-NINsMigrated	Accumulated number of INs migrated.
je-env-stat-NINsObsolete	Accumulated number of INs obsolete.
je-env-stat-NLNQueueHits	Accumulated number of LNs processed without a tree lookup.
je-env-stat-NLNsCleaned	Accumulated number of LNs cleaned.
je-env-stat-NLNsDead	Accumulated number of LNs that were not found in the tree anymore (deleted).
je-env-stat-NLNsLocked	Accumulated number of LNs encountered that were locked.
je-env-stat-NLNsMarked	Accumulated number of LNs that were marked for migration during cleaning.
je-env-stat-NLNsMigrated	Accumulated number of LNs migrated.
je-env-stat-NLNsObsolete	Accumulated number of LNs obsolete.
je-env-stat-NLogBuffers	Number of log buffers currently instantiated.
je-env-stat-NMarkedLNsProcessed	Accumulated number of LNs processed because they were previously marked for migration.
je-env-stat-NNodesExplicitlyEvicted	Accumulated number of nodes evicted.

Monitor Attribute	Description
je-env-stat-NNodesScanned	Accumulated number of nodes scanned to select the eviction set.
je-env-stat-NNodesSelected	Accumulated number of nodes selected to evict.
je-env-stat-NNotResident	Number of requests for database objects not contained within the in memory data structures.
je-env-stat-NOpenFiles	Number of files currently open in the file cache.
je-env-stat-NPendingLNLocked	Accumulated number of pending LNs that could not be locked for migration because of a long duration application lock.
je-env-stat-NPendingLNProcessed	Accumulated number of LNs processed because they were previously locked.
je-env-stat-NRandomReadBytes	Number of bytes read which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomReads	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWriteBytes	Number of bytes written which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWrites	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NRepeatFaultReads	Number of reads which had to be repeated when faulting in an object from disk because the read chunk size controlled by <code>je.log.faultReadSize</code> is too small.
je-env-stat-NRepeatIteratorReads	Number of times we try to read a log entry larger than the read buffer size and can't grow the log buffer to accommodate the large object.
je-env-stat-NRootNodesEvicted	Accumulated number of database root nodes evicted.
je-env-stat-NSequentialReadBytes	Number of bytes read which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialReads	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWriteBytes	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWrites	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NSharedCacheEnvironments	Number of environments using the shared cache.
je-env-stat-NTempBufferWrites	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by <code>je.log.totalBufferBytes</code> and <code>je.log.numBuffers</code> were not large enough.
je-env-stat-NToBe-CleanedLNProcessed	Accumulated number of LNs processed because they are soon to be cleaned.
je-env-stat-NonEmptyBins	Number of non-empty bins.
je-env-stat-ProcessedBins	Number of bins that were successfully processed by the INCompressor.
je-env-stat-RequiredEvictBytes	Number of bytes that must be evicted to get within the memory budget.
je-env-stat-SharedCacheTotalBytes	Total amount of the shared JE cache in use, in bytes.
je-env-stat-SplitBins	Number of bins encountered by the INCompressor that were split between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-TotalLogSize	Approximation of the current total log size in bytes.
je-env-stat-NOwners	Total lock owners in the lock table.

Monitor Attribute	Description
je-env-stat-NReadLocks	Total read locks currently held.
je-env-stat-NRequests	Total number of lock requests to date.
je-env-stat-NTotalLocks	Total locks currently in the lock table.
je-env-stat-NWaiters	Total transactions waiting for locks.
je-env-stat-NWaits	Total number of lock waits to date.
je-env-stat-NWriteLocks	Total write locks currently held.
je-env-stat-LastCheckpointTime	Time of the last checkpoint.
je-env-stat-LastTxnId	Last transaction ID allocated.
je-env-stat-NAborts	Number of transactions that have aborted.
je-env-stat-NActive	Number of transactions that are currently active.
je-env-stat-NBegins	Number of transactions that have begun.
je-env-stat-NCommits	Number of transactions that have committed.
je-env-stat-NXAAborts	Number of XA transactions that have aborted.
je-env-stat-NXACommits	Number of XA transactions that have committed.
je-env-stat-NXAPrepares	Number of XA transactions that have been prepared.

Summary of the Replication Summary Monitor Information

The following table provides a description of the attributes in the `cn=Replication Summary` monitor entry.

```
dn: cn=Replication Summary <baseDN>,cn=monitor
```

Table 55: Replication Summary Monitor Information

Monitor Attribute	Description
base-dn:<baseDN>	Base DN summary.
replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest-backlog-change	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.
replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective <code>serverID</code> and <code>server</code> properties.
update-queue: id, max-count, current-count, max-size, current-size, polling-source, polling-source-changed	Summary information for each update queue on a server: <ul style="list-style-type: none"> > id. The ID of the receiving replica or replication server > max-count. The maximum number of update messages that the sending replication server will keep in memory for the receiving replica or replication server. If the receiver cannot accept messages fast enough for any reason (high load, network latency, etc), then this queue will fill up. When that happens, the sending replication server will read update messages from the changelog backend. This slows down the update processing considerably. > current-count. The number of update messages currently on the queue that have not been sent to the receiving replica or replication server.

Monitor Attribute	Description
	<p>Every time the sending replication server sends an update to the receiving replica or replication server, this counter is decremented.</p> <ul style="list-style-type: none"> > max-size. The maximum total size (in bytes) of update messages that may be in the queue. This queue is capped by both the maximum count (<code>max-count</code>) and the <code>max-size</code> setting, whichever is reached first. > current-size. The total size of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this value is decremented by the size of the published update message. > polling-source. Either 'memory' or 'db'. If set to 'memory', the sending replication server relies only on the in-memory queue to push update messages to the receiving replica or replication server. If set to 'db', update messages are read and sorted from the changelog database, which is significantly slower than publishing updates from the in-memory queue. > polling-source-changed. The total number of times the polling-source attribute has changed value (either from 'db' to 'memory' or from 'memory' to 'db'). If this value changes very frequently, then the queue size setting is probably too low.

Summary of the replicationChanges Backend Monitor Information

The following table provides a description of the attributes in the `cn=replicationChanges` Backend monitor entry.

```
dn: cn=replicationChanges Backend,cn=monitor
```

Table 56: replicationChanges Backend Monitor Information

Monitor Attribute	Description
<code>ds-backend-id</code>	ID descriptor for the backend. Typically, this will be "replicationChanges".
<code>ds-backend-base-dn</code>	Base DN for the backend. Typically, this will be <code>cn=replicationChanges</code> .
<code>ds-backend-is-private</code>	Flag to indicate if the backend is private.
<code>ds-backend-entry-count</code>	Entry count for the backend.
<code>ds-base-dn-entry-count</code>	Entry count for the base DN and the specified base DN.
<code>ds-backend-writability-mode</code>	Flag to indicate if the backend is writable or not.

Summary of the Replication Protocol Buffer Monitor Information

The following table provides a description of the attributes in the `cn=Replication Protocol Buffer` monitor entry. The monitors provide information on the state of the buffer for protocol operations, which is kept in the local storage.

```
dn: cn=Replication Protocol Buffer,cn=monitor
```

Table 57: Replication Protocol Buffer Monitor Information

Monitor Attribute	Description
saved-buffers	Number of protocol buffers. Initial buffer size is 4k.
reallocations	Number of times the buffers had to be reallocated due to insufficient size.
large-buffer-creates	Number of times a buffer larger than 512k was requested.
large-buffer-evictions	Number of times a buffer was removed from the thread local storage, because it has grown above 512k.

Advanced Topics Reference

This chapter presents background reference information covering advanced replication topics.

About the Replication Protocol

Replication communicates using a proprietary binary protocol that is implemented on top of the TCP/IP protocol using SSL encryption. Some protocol messages are used for administrative purposes (such as WAN Gateway server negotiation or flow control), some carry updates to replicated data, while others are directed to all servers for monitoring requests.

In a replicated topology, each participating Directory Server is connected to every other server via the replication server port in order to monitor health. Servers which share the same location setting are also connected to rapidly replicate changes and lastly the WAN Gateway servers are all interconnected to replicate changes across locations.

Directory Servers keep connections open as long as possible to reduce the communication latency when messages are exchanged. Heartbeat messages are transmitted on a regular basis to detect a network failure or an unresponsive directory server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

The following detailed communication flow will be used to describe major components of replication. This illustration is the expanded view of figure shown in the Overview section.

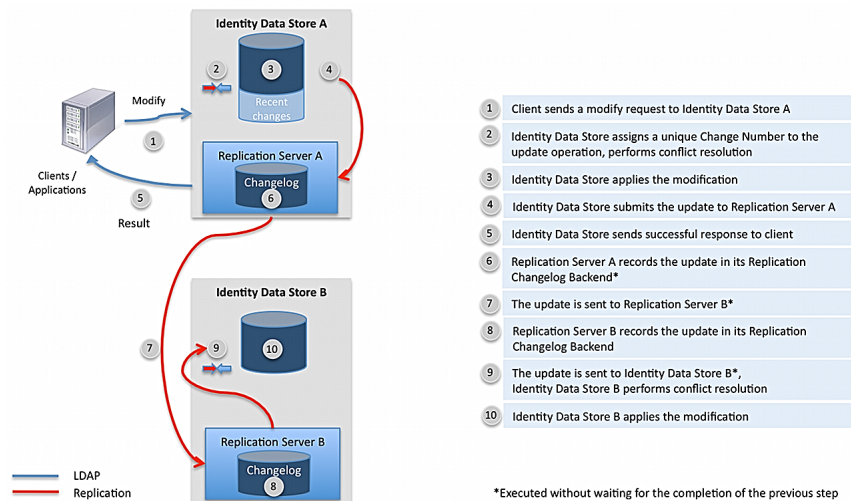


Figure 16: Replication Communication Flow

Step 1. Client sends a Modify request to Directory Server A.

Step 2. Directory Server A assigns a unique change number to the operation. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry. The change number is assigned before the Directory Server backend is updated so that the arrival order of client requests can be preserved. Historical data in the target entry is updated to reflect the change. Note that historical data is only updated for ADD and MODIFY operations.

Step 3. Directory Server applies the modifications in the corresponding backend.

Step 4. If the MODIFY operation successfully completes, then the Directory Server will submit the update to its embedded *Replication Server*. The Replication Server is a component within the Directory Server process responsible for propagating updates to and from the replication topology. The Directory Server itself only communicates with a single replication server, whereas the replication server component is connected to all other replication servers. If the Directory Server process exits unexpectedly and some updates have not been passed to the Replication Server, the backend has the ability to recover the last 50,000 recent changes that were processed at this server, guaranteeing that these changes can be replicated when the server starts up. The figure above also shows that replication protocol is used not just between replication servers but also between the Directory Server and the Replication Server.

Step 5. The response is sent to the client. In this example, a successful response is assumed.

Step 6. The Replication Server records the update in its own Changelog backend (i.e., backend ID of `replicationChanges`) and on disk with the path to `changelogDb` under the server root. The Replication Changelog backend keeps track of updates at each and every Directory Server in the replication topology. When a Directory Server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this Directory Server. Old records from the Replication Changelog backend are purged, which by default removes records older than 24 hours. If the backend does not contain all of the records that another Directory Server needed when rejoining the replication topology, then the replicated data set in the Directory Server must be re-initialized. In this case, the Directory Server enters lockdown mode and an administrative alert is sent.

Step 7. The Replication Server submits the update to the replication server component in Directory Server B. If there were more Directory Servers in this example, the Replication Server would submit the update to all the other replication servers in the same location.

Step 8. Just like in Step 6, the Replication Server component receiving an update inserts the change into its Replication Changelog backend.

Step 9. The update is forwarded to the Replica in Directory Server B. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry.

Step 10. The Directory Server applies the modification in the corresponding backend. The Recent Changes database is not updated, because only updates that originated at this Directory Server are recorded in the Recent Changes database.

Change Number

As seen in the previous Figure, the Directory Server assigns a unique change number to each update operation (specifically, ADD, DELETE, MODIFY, or MODIFY DN operations) to track each request when received from a client. The change number not only identifies each and every update, but it also allows ordering updates to the replicated data the same way on each Directory Server. The change number is composed of the following multiple fields:

- **Timestamp** that identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.
- **Server ID** that uniquely identifies the Directory Server where the update was made.
- **Sequence number** that defines the order of the updates received from external clients at a particular directory server.

The replication protocol also sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, however, it is still recommended to keep the system clocks synchronized.

Conflict Resolution

The eventual-consistency model employed in replication introduces a window where conflicting updates targeting the same entry may be applied at two different Directory Servers. In general, two updates to the same Directory Server are in conflict if the update that arrived later fails. Conflict resolution, when possible, corrects conflicts introduced by clients automatically. There are some exceptions, however, when manual administrative action is required. For example, adding an entry in one replica and deleting the parent of this entry on another replica simultaneously will introduce a conflict that requires manual action. In a carefully implemented deployment, the risk of introducing conflicts that require manual action can be significantly reduced or even eliminated.

The conflict resolution algorithm in the Ping Identity Directory Server uses a mechanism that orders all updates in the replication topology. Each update in the Directory Server is assigned a

unique change number. The change number is attached to each update propagated via replication and allows each Directory Server to order updates exactly the same way.

Consider the following example that results in a conflict: add a single-valued attribute with different values to an entry concurrently at two Directory Servers (shown in the figure below). It is easy to see that the second operation would fail if a client attempted to add the same attribute to the same entry at the same Directory Server. In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different Directory Servers. The conflict is handled only after replication propagates the updates. The Directory Servers resolve the conflict independently of the other server. On one Directory Server, the entry will be updated to reflect the correct value; on the other Directory Server, the value will stay the same. As result, each Directory Server will independently resolve the conflict the same way based on the ordering of the updates. This example is illustrated below:

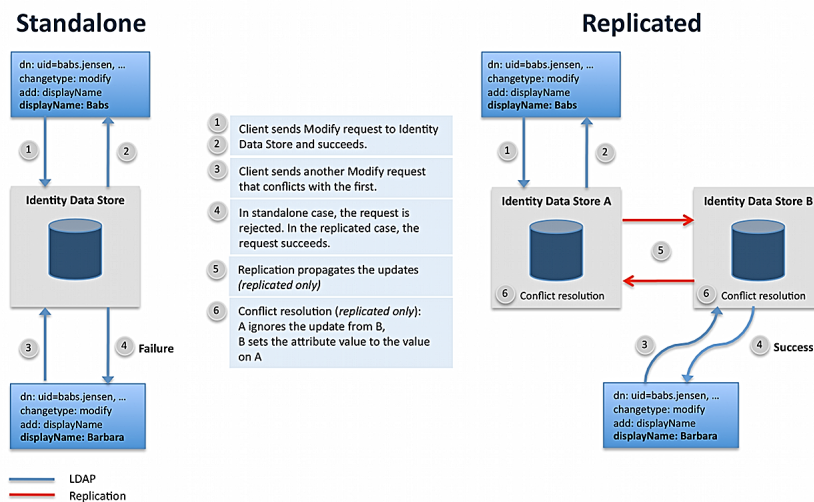


Figure 17: Conflict Resolution Process Flow

WAN-Friendly Replication

Many multi-national corporations that have data centers in different countries must minimize latency over WAN to ensure acceptable performance for their client applications. To minimize WAN latency, the Directory Server assigns one of two roles to the replication servers: the role of standard replication transmitting updates to the other co-located replication servers; the other role, a WAN-dedicated replication server designed to send updates to other WAN-designated replication servers in other locations. This two-role system minimizes WAN traffic by pushing all replication updates onto the connected replication servers that are designated as WAN Gateway Servers. Only the designated WAN Gateway Servers can transmit the update messages to other connected WAN Gateway servers at other locations.

WAN Gateway Server

The Directory Server's replication mechanism relies on the server's location information to reduce protocol traffic on WAN links. During protocol negotiation, the replication server with the highest WAN Gateway priority (priority 1 indicates the highest priority) automatically assumes the role as the WAN Gateway Server for that particular location. The Gateway Server's

main function is to route update messages from other non-gateway servers at the same location to remote WAN Gateway servers at other locations. Similarly, at the destination point, the replication server with the WAN Gateway role will receive update messages from other WAN gateway servers at other locations and push them out to all replication servers at the current location. This setup ensures that all WAN communication flows through the WAN Gateway Servers.

The figure below shows a basic connection configuration for updates. Keep in mind that all of the replication servers are fully connected to each other for monitoring or server negotiation purposes.

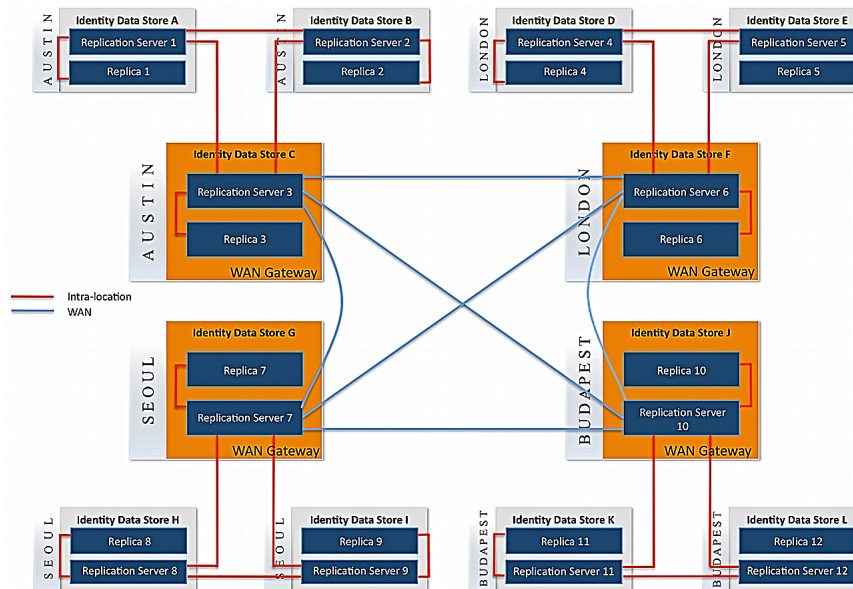


Figure 18: WAN Gateway Servers

If the WAN Gateway Server is temporarily unavailable due to a planned or unplanned downtime, the system will dynamically re-route updates to a newly designated WAN Gateway Server in the same location. The replication server with the next highest WAN Gateway priority number automatically assumes the WAN Gateway role. For deployments with entry-balancing Directory Proxy Servers, there will be one WAN Gateway Server per data set.

By default, all servers are enabled to serve as WAN Gateways and all are set to priority 5, which is simply a way to make them all equal. If necessary, the WAN Gateway priority can be changed using `dsconfig` after replication has been enabled.

WAN Message Routing

Non-gateway replication servers forward update messages from replicas to co-located replication servers only. It is the responsibility of the WAN gateway server to forward these messages to WAN gateway servers at other locations. The figure below illustrates how an update message is routed from a non-gateway server to a remote location.

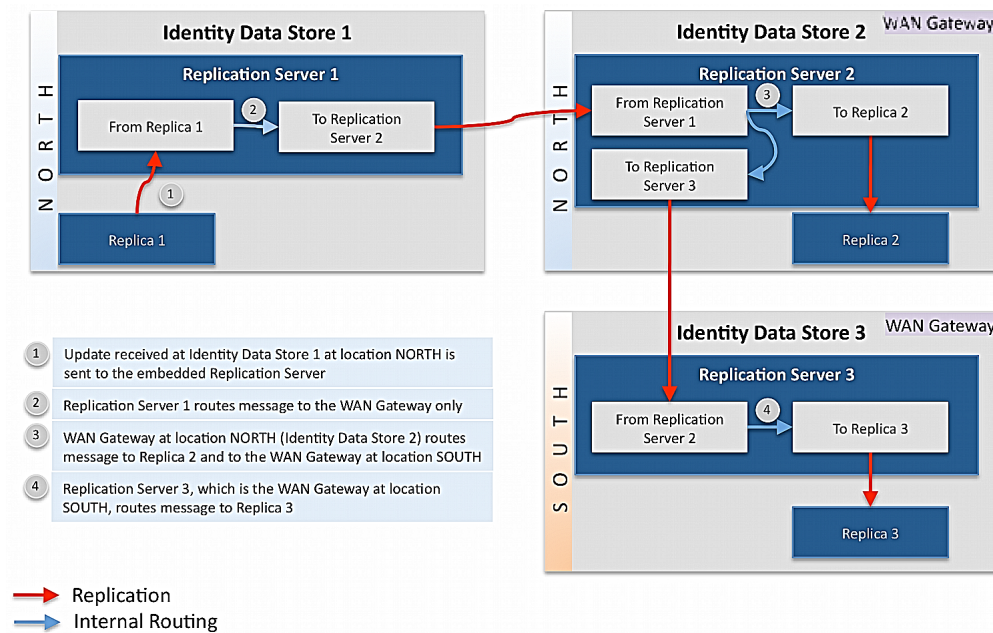


Figure 19: WAN Message Routing

WAN Gateway Server Selection

The WAN Gateway role is dynamically assigned to the most suitable server in a particular location. In most cases, the replication server with the higher WAN Gateway priority (e.g., priority 1 indicates the highest priority) assumes this role.

A replication server will not attempt to become a WAN Gateway if any of the following conditions exists:

- > The WAN Gateway priority is set to 0
- > The replication server is backlogged at server startup
- > The current WAN Gateway has higher priority
- > The WAN Gateway has not been elected yet, but higher priority replication servers are present in the location

The currently active WAN Gateway server will give up the gateway role in the following cases:

- > The server has started the shutdown process
- > The server is preparing for a scheduled maintenance cycle
- > The server learns about a higher priority replication server in its location

Replication servers send WAN Gateway information to other servers at regular intervals using the replication protocol. This allows the replication servers to take the appropriate action, for example, to become a WAN Gateway, if necessary without any manual administrative action.

WAN Replication in Mixed-Version Environments

WAN Gateway Role assignment is only available on Ping Identity Directory Servers version 3.5 or later. Legacy servers (i.e., pre-3.5) will never be selected as WAN Gateways. Updates from legacy replication servers are forwarded to all replication servers and not funneled to a single WAN Gateway. Also, updates from remote WAN Gateways will be forwarded to the legacy replication servers. Both of these actions will effectively increase WAN traffic during replication.

Recovering a Replication Changelog

In the event that the replication changelog is compromised (`<server-root>/changelogDb`), possibly due to a disk or NAS failure, perform the following steps.

1. Stop the server.
2. Backup `replicationChanges` from a remote server with the following command:

```
$ bin/backup --backupDirectory /app/backups/replicationChanges \  
--backendID replicationChanges
```

3. Copy the `replicationChanges` backup from the remote server and restore it on the local host as follows:

```
$ bin/restore --backupDirectory /app/tmp/replicationChanges
```

4. Start the server.

Disaster Recovery

In the event that data is compromised across all systems and a restore is necessary, perform the following steps. These steps assume that no read or write operations are performed by any servers during this process.

1. Stop all servers.
2. Run the following command on all servers:

```
$ /bin/dsreplication cleanup-local-server
```

3. Locate the backup or exported LDIF file that represents the last working copy of the database.
4. Restore the backup or import the LDIF file on a single server. If importing an LDIF file, use the `--excludeReplication` option with the `import-ldif` command.
5. Start the restored server. The server can now receive client requests.

6. Start another server in lockdown mode with the following command:

```
$ start-ds --skipPrime --lockdownMode
```

7. Enable replication from the first server to the second server.

8. Initialize the second server from the first with the following command:

```
$ bin/dsreplication initialize
```

9. Restart the second server or use the `bin/leave-lockdown-mode` command to leave the server in lockdown mode. The second server can now receive client requests.

10. Repeat steps 6 through 9 for any other servers.

Chapter

20

Managing Logging

The Ping Identity Directory Server supports a rich set of log publishers to monitor access, debug, and error messages that occur during normal server processing. Administrators can view the standard set of default log files as well as configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

This chapter presents the following information:

Topics:

- [Default Directory Server Logs](#)
- [Types of Log Publishers](#)
- [Managing Access and Error Log Publishers](#)
- [Managing File-Based Access Log Publishers](#)
- [Generating Access Logs Summaries](#)
- [About Log Compression](#)
- [About Log Signing](#)
- [Creating New Log Publishers](#)
- [Configuring Log Rotation](#)
- [Configuring Log Rotation Listeners](#)
- [Configuring Log Retention](#)
- [Configuring Filtered Logging](#)
- [Managing Admin Alert Access Logs](#)
- [Managing Syslog-Based Access Log Publishers](#)
- [Managing the File-Based Audit Log Publishers](#)
- [Managing the JDBC Access Log Publishers](#)
- [Managing the File-Based Error Log Publisher](#)
- [Managing the Syslog-Based Error Log Publisher](#)
- [Creating File-Based Debug Log Publishers](#)

Default Directory Server Logs

The Directory Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the `PingDirectory/logs` directory. The following default log files are available on the Directory Server and are presented below.

Table 58: Directory Server Logs

Log File	Description
access	File-based Access Log that records operations processed by the Directory Server. Access log records can be used to provide information about problems during operation processing (for example, unindexed searches, failed requests, etc.), and provide information about the time required to process each operation.
change-notifications.log	Records changes to data anywhere in the server, which match one or more configured change subscriptions.
config-audit.log	Records information about changes made to the Directory Server configuration in a format that can be replayed using the <code>dsconfig</code> tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
expensive-ops	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
failed-ops	Failed Operations Log. Provides information on all operations that failed in single-line log format.
replication	Records any replication errors and publishes them to the filesystem.
searches-returning-no-entries	Searches Returning No Entries Log. Disabled by default. Provides information only on operations that did not return any entries.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to <code>server.out</code> .
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a Directory Server with the <code>setup</code> tool.
tools	Directory that holds logs for long running utilities: <code>import-ldif</code> , <code>export-ldif</code> , <code>backup</code> , <code>restore</code> , <code>verify-index</code> . Current and previous copies of the log are present in the Directory Server.
update.log	Records messages that occur during a Directory Server update.

Types of Log Publishers

The Ping Identity Directory Server provides several classes of log publishers for parsing, aggregating, and filtering information events that occur during normal processing in the server. There are three primary types of Log Publishers: access logs, debug logs, and error logs. Each type has multiple subtypes of log files based on the log server type:

Table 59: Types of Log Publishers

Log Publisher Type	Description	
Access	Provides the requests received and responses returned by the Directory Server. The information can be used to understand the operations performed by clients and to debug problems with the client applications. It can also be used for collecting usage information for performance and capacity planning purposes. There are tools described later that can analyze the access log to provide summaries of the LDAP activity and performance.	
	File-based Audit Log	Special type of access logger that provides detailed information about changes processed within the server. Disabled by default.
	JDBC-based Access Log	Stores access log information using a JDBC database connection. Disabled by default.
	File-based Access Logs	<p>Provides a character-based stream used by TextWriter Publishers as a target for outputting log records. There are six types of file-based loggers:</p> <ul style="list-style-type: none"> • Admin Alert Access Log. Generates administrative alerts for any operations that match the criteria for this access logger. Disabled by default. • File-based Access Log. Publishes access log messages to the filesystem. Enabled by default. • Syslog-based Access Log. Publishes access log messages to a syslogd port. Disabled by default. • Expensive-Operations Access Log. Publishes only those access log messages of operations that take longer than 1000 milliseconds. Disabled by default. • Failed-Operations Access Log. Publishes only those access log messages of operations that failed for any reason. Enabled by default. • Successful Searches with No Entries Returned Log. Publishes only those access log messages of search operations that failed to return any entries. Disabled by default.
Debug	Provides information about warnings, errors, or significant events that occur within the server.	
	Debug ACI Logger	Stores debug information on ACI evaluation for any request operations against the server
	File-based Error Logs	<p>There are two types of File-based Error Logs:</p> <ul style="list-style-type: none"> • Error log. Publishes error messages to the filesystem. Enabled by default. • Replication log. Publishes replication error messages to the filesystem. Enabled by default.
	JDBC-based Error Logs	Stores error log information using a JDBC database connection. Disabled by default.
	Syslog-based Error Logs	Publishes error messages to a syslogd port.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Directory Server using the `dsconfig` tool.



Note: Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using `dsconfig` before they appear in the list of log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

- Use `dsconfig` to view the log publishers.

```
$ bin/dsconfig list-log-publishers
```

Log Publisher	Type	enabled
Debug ACI Logger	debug-access	false
Expensive Operations Access Logger	file-based-access	false
Failed Operations Access Logger	file-based-access	true
File-Based Access Logger	file-based-access	true
File-Based Audit Logger	file-based-audit	false
File-Based Debug Logger	file-based-debug	false
File-Based Error Logger	file-based-error	true
Replication Repair Logger	file-based-error	true
Successful Searches with No Entries Returned	file-based-access	false

Enabling or Disabling a Default Log Publisher

You can enable or disable any log publisher available on the Directory Server using the `dsconfig` tool. By default, the following loggers are disabled and should be enabled only when troubleshooting an issue on the server:

- > Expensive Operations Access Logger
- > File-Based Audit Logger
- > File-Based Debug Logger
- > Successful Searches with No Entries Returned

To Enable a Default Access Log

- Use `dsconfig` to enable an Access Log Publisher. In this example, enable the Expensive-Ops log, which will record only those access log messages that take 1000 milliseconds or longer.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Expensive Operations Access Logger" --set enabled:true
```

Managing Access and Error Log Publishers

The Access Log records every request received and response returned by the Directory Server. The Access Log stores the IDs for the client connection, operation, the LDAP message involved with each client request, and the server response. The information can be used to debug any problems with a client application by correlating the numeric operation identifier to the client request or response.

The Directory Server supports multiple classes of access log publishers depending on your logging requirements. The following types of access log publishers are available on the system:

- **File-Based Access Log Publishers.** Provides a character-based TextWriter stream for outputting log records. There are three subclasses of TextWriter access logs:
 - **File-Based Access Logs.** Enabled by default. The File-based Access Log publishes access messages to the filesystem as `<server-root>/logs/access`. The Failed-Operations Log, Expensive-Operations Log, and the Searches with No Entries Returned Log are specialized types of the File-Based Access Log and shows only specific information necessary for troubleshooting purposes.
 - **Admin-Alert Access Logs.** Disabled by default. The Admin-Alert Access Log is specialized type of logger that automatically generates administrative alerts for any operations that match a criteria for this access log publisher.
 - **Syslog-Based Access Logs.** Disabled by default. The Syslog Access Log publishes access messages to a syslogd port.
- **File-Based Audit Logs.** Disabled by default. The Audit Log provides detailed information about modifications (writes) processed within the Directory Server. The File-based Audit Log publishes access messages to the filesystem as `<server-root>/logs/audit`.
- **JDBC-Based Access Logs.** Disabled by default. The JDBC-based Access Log provides information using a JDBC database connection.
- **JDBC-Based Error Logs.** Disabled by default. The JDBC-based Error Log provides information using a JDBC database connection.

Managing File-Based Access Log Publishers

The Ping Identity Directory Server supports a flexible and configurable Access Logging system that provides a full set of customized components to meet your system's logging requirements. The default Access Log can be configured to write information to a log file with two records per operation, one for the request received and one for the response returned. It can also be configured to write one message per operation or configured to record information about a subset of operations processed by the server. In addition to modifying existing default log files, you can create custom log publishers to monitor specific properties or connection criteria. For more information, see [Creating New Log Publishers](#).

The Directory Server can be configured to use multiple access log publishers writing logs to different files using different configurations. This approach makes it possible to have fine-grained logging for various purposes (for example, a log that contains only failed operations, a log that contains only operations requested by root users, or a log that contains only operations that took longer than 20ms to complete).

The Directory Server provides an additional mechanism to filter access logs to record only a subset of messages of one or more types. The access log filtering mechanism uses the operation criteria (connection, request, result, search-entry, search-reference) to determine whether a given message should be logged based on information associated with the client connection as well as information in the operation request/response messages. For more information, see [Configuring Filtered Logging](#).

Access Log Format

The Access Log has a standard format that lists various elements identifying the connection and operation occurring within the Directory Server. By default, each operation generates one access log message.

The Access Log displays the following common properties:

- **Timestamp.** Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>
- **Connection Type.** Displays the connection type requested by the client and the response by the server. Examples include the following:
 - > CONNECT
 - > BIND REQUEST/RESULT
 - > UNBIND REQUEST
 - > DISCONNECT
 - > SEARCH REQUEST/RESULT
 - > MODIFY REQUEST/RESPONSE
 - > others include: ABANDON, ADD, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFY DN
- **Connection ID.** Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation. The connection ID is unique for a span of time on a single server. Values of the connection ID will be re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.
- **Operation ID.** Numeric identifier, starting incrementally with 0, that identifies the operation. The operation ID is unique for a span of time on a single server. Values of the operation ID will be re-used when the server restarts or when it has serviced enough operations to cause the identifier to wrap back to zero.
- **Result Code.** LDAP result code that determines the success or failure of the operation result. Result messages include a result element that indicates whether the operation was successful or if failed, the general category for the failure, and an etime element that indicates the length of time in milliseconds that the server spent processing the operation.

The Directory Server provides a useful tool `<server-root>/bin/ldap-result-code` (UNIX, Linux) or `<server-root>\bat\ldap-result-code` (Windows), that displays all of

the result codes used in the system. You can use the utility if you are not sure what a result code means. For example, use the following:

- `ldap-result-code --list` displays all of the defined result codes in the Directory Server.
- `ldap-result-code --int-value 16654` displays the name of the result code with a numeric value of 16654.
- `ldap-result-code --search operation` displays a list of all result codes whose name includes the substring "operation".
- **Elapsed Time.** Displays the elapsed time (milliseconds) during which the operation completed its processing.
- **Message ID.** Numeric identifier, starting incrementally with 1, which identifies the LDAP message used to request the operation.

Access Log Example

The following example shows output from the Access Log in `<server-root>/logs/access`:

```
[01/Jun/2011:14:48:17 -0500] CONNECT conn=0 from="10.8.1.243" to="10.8.1.243"
protocol="LDAP"
[01/Jun/2011:14:48:17 -0500] BIND REQUEST conn=0 op=0 msgID=1 version="3"
dn="cn=Directory Manager"
authType="SIMPLE"
[01/Jun/2011:14:48:17 -0500] BIND RESULT conn=0 op=0 msgID=1 resultCode=0 etime=26.357
authDN="cn=Directory Manager,cn=Root DNs,cn=config"
[01/Jun/2011:14:48:17 -0500] UNBIND REQUEST conn=0 op=1 msgID=2
[01/Jun/2011:14:48:17 -0500] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the Access Log Using dsconfig Interactive Mode

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the `dsconfig` tool in interactive or non-interactive mode.

To Modify the Access Log Using dsconfig Interactive Mode

1. Use `dsconfig` in interactive mode to modify the access log properties.

```
$ bin/dsconfig
```

2. Follow the prompts to specify the LDAP connection parameters for host name or IP address, connection type (LDAP, SSL, or StartTLS), port number, bind DN and password.
3. On the Directory Server main menu, type the number corresponding to the Log Publisher.
4. On the **Log Publisher Management** menu, enter the option to view and edit an existing log publisher.
5. On the **Log Publisher** menu, type the number corresponding to File-based Access Logger.

6. On the **File-Based Access Log Publisher** menu, type the number corresponding to the property that you want to change, and then follow the prompts.
7. Type `f` to apply the changes.

Modifying the Access Log Using dsconfig Non-Interactive Mode

You can use the `dsconfig` tool in non-interactive mode to quickly modify a log publisher property on the command line or in a script. For information on each property, see the *Directory Server Configuration Reference Guide*, which is an HTML document listing the various properties for each Directory Server component.

To Modify the Access Log Using dsconfig Non-Interactive Mode

- Use `dsconfig` with the `--no-prompt` option with the properties that you want to modify or set for your access log. In this example, enable the properties to include the instance name and startup ID.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Access Logger" \  
--set include-instance-name:true --set include-startup-id:true
```

Modifying the Maximum Length of Log Message Strings

By default, the Directory Server sets the maximum length of log message strings to 2000 characters. This value is configurable for any access log publisher, except the Syslog publisher, which is set to 500 characters. You can change the maximum length of log message strings by setting the `max-string-length` configuration property. If any string has more than the configured number of characters, then that string will be truncated and a placeholder will be appended to indicate the number of remaining characters in the original string.

To Modify the Maximum Length of Log Message Strings

- Use `dsconfig` to set the `max-string-length` property for an access log. The following command configures the "File-based Access Logger" to include the instance name and the maximum length of the log message strings to 5000 characters.

```
$ bin/dsconfig set-log-publisher-prop \  
--publisher-name "File-Based Access Logger" \  
--set include-instance-name:true \  
--set max-string-length:5000
```

Generating Access Logs Summaries

The Directory Server provides a convenience tool, `summarize-access-log`, that generates a summary of one or more file-based access logs. The summary provides analytical metric information that could be useful for administrators. The following metrics are provided in each summary:

- Total time span covered by the log files.
- Number of connections established and the average rate of new connections per second.
- IP addresses of up to the top 20 of the clients that most frequently connect to the server, the number of connections by each address, and the percentage of connections of each.
- Total number of operations processed, the number of operations of each type, the percentage of each type out of the total number, and the average rate per second for each type of operation.
- Average processing time for all operations and for each type of operation.
- Histogram of the processing times for each type of operation.
- Up to the 20 most common result codes for each type of operation, the number of operations of that type with that result code, and the percentage of operations of that type with that result code.
- Number of unindexed searches processed by the server.
- Breakdown of the scopes used for search operations with the number of percentage of searches with each scope.
- Breakdown of the most common entry counts for search operations with the number and percentage of searches that returned that number of entries.
- Breakdown of the most commonly used filter types for searches with a scope other than "base" (that is, those searches for which the server will use es when processing the filter). These filters will be represented in a generic manner so that any distinct assertion values or substring assertion elements will be replaced with question marks and attribute names in all lowercase characters (for example, (givenName=John) would be represented as (givenName=?)).

To Generate an Access Log Summary

- Use the `bin/summarize-access-log` with path to one or more access log files.

```
$ bin/summarize-access-log /path/to/logs/access

Examining access log /path/to/logs/access Examined 500 lines in 1 file covering a
total duration of 1 day, 22 hours, 57 minutes, 31 seconds

Total connections established: 69 (0.000/second)
Total disconnects: 69 (0.000/second)

Most common client addresses:
127.0.0.1: 61 (88.406)
10.8.1.209: 8 (11.594)

Total operations examined: 181 ...
(metric for each operation examined) ...

Average operation processing duration: 22.727ms
Average add operation processing duration: 226.600ms
Average bind operation processing duration: 5.721ms
Average delete operation processing duration: 77.692ms
Average modify operation processing duration: 35.530ms
Average search operation processing duration: 4.017ms

Count of add operations by processing time:
```

```

... (histogram for add operations) ...

Count of bind operations by processing time:
... (histogram for bind operations) ...

Count of delete operations by processing time:
... (histogram for delete operations) ...

Count of modify operations by processing time:
... (histogram for modify operations) ...

Count of search operations by processing time:
... (histogram for search operations) ...

Most common add operation result codes:
success: 11 (84.615%)
entry already exists: 2 (15.385%)

Most common bind operation result codes:
success: 4 (50.000%)
invalid credentials: 4 (50.000%)

Most common delete operation result codes:
success: 1 (100.000%)

Most common modify operation result codes:
success: 9 (69.231%)
no such object: 4 (30.769%)

Most common search operation result codes:
success: 133 (91.724%)
no such object: 12 (8.276%)

Number of unindexed searches: 0

Most common search scopes:
BASE: 114 (78.621%)
SUB: 16 (11.034%)
ONE: 15 (10.345%)

Most common search entry counts:
1: 119 (82.069%)
0: 17 (11.724%)
2: 5 (3.448%)
10: 4 (2.759%)

Most common generic filters for searches with a non-base scope:
(objectclass=?): 19 (61.290%)
(ds-backend-id=?): 12 (38.710%)

```

About Log Compression

The Directory Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The `summarize-access-log` tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting

purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About Log Signing

The Directory Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be properly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the `dsconfig` tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the `validate-file-signature` tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

To Configure Log Signing

1. Use `dsconfig` to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit Logger" \
--set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit Logger" \
--set enabled:false
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit Logger" \
--set enabled:true
```

To Validate a Signed File

The Directory Server provides a tool, `validate-file-signature`, that checks if a file has not been tampered with in any way.

- Run the `validate-file-signature` tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
All signature information in file 'logs/audit' is valid
```

Note: If any validation errors occur, you will see a message similar to the one as follows:



```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

Creating New Log Publishers

The Ping Identity Directory Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see [Configuring Log Rotation and Configuring Log Retention](#).

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
--type file-based-access --publisher-name "Disconnect Logger" \
--set enabled:true \
--set "rotation-policy:24 Hours Time Limit Rotation Policy" \
--set "rotation-policy:Size Limit Rotation Policy" \
--set "retention-policy:File Count Retention Policy" \
--set log-connects:false \
--set log-requests:false --set log-results:false \
--set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:



```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring Log Rotation

The Directory Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy.** Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy.** Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy.** Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy.** Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \  
  --publisher-name "File-Based Access Logger" \  
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \  
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Rotation Listeners

The Directory Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \  
  --listener-name "Copy on Rotate" \  
  --type copy \  
  --set enabled:true \  
  --set copy-to-directory:/path/to/archive/directory \  
  --set compress-on-copy:true
```

The path specified by the `copy-to-directory` property must exist, and the filesystem containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target filesystem and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the `summarize-access-log` tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the `summarize-access-log` tool, including the `file-based-access` and `operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \  
  --listener-name "Summarize on Rotate" \  
  --type summarize \  
  --set enabled:true \  
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there

is enough space available in the specified storage directory. The server automatically monitors free disk space on the filesystem to which the summary files are written.

Configuring Log Retention

The Directory Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy.** Sets the number of log files you want the Directory Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy.** Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy.** Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy.** Create a new retention policy that meets your Directory Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy.** Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \  
  --publisher-name "File-Based Access Logger" \  
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring Filtered Logging

The Ping Identity Directory Server provides a mechanism to filter access log messages based on specific criteria. The filtered log can then be used with a custom log publisher to create and to generate your own custom logs. Adding new filtered logs and associate publishers does not change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful does not result in those operations being removed from the default access log.

The following example shows how to create a set of criteria that matches any operation that did not complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria. Note that this log does not include messages for connects

or disconnects, and only a single message is logged per operation. This message contains both the request and result details.

To run log filtering based on any operation result (for example, result code, processing time, and response controls), turn off request logging and set the `include-request-details-in-result-messages` property to `TRUE`. Since filtering based on the results of an operation cannot be done until the operation completes, the server has no idea whether to log the request. Therefore, it might log request messages but not log any result messages. Instead, if you can only log result messages and include request details in the result messages, then only messages for operations that match the result criteria are logged. All pertinent information about the corresponding requests are included.

To Configure a Filtered Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create a result criteria object set to `failure-result-codes`, a predefined set of result codes that indicate when an operation did not complete successfully.

```
$ bin/dsconfig create-result-criteria --type simple \  
  --criteria-name "Failed Operations" --set result-code-criteria:failure-result-codes
```

2. Use `dsconfig` to create the corresponding log publisher that uses the result criteria. The log rotation and retention policies are also set with this command.

```
$ bin/dsconfig create-log-publisher \  
  --type file-based-access \  
  --publisher-name "Filtered Failed Operations" \  
  --set enabled:true \  
  --set log-connects:false \  
  --set log-disconnects:false \  
  --set log-requests:false \  
  --set "result-criteria:Failed Operations" \  
  --set log-file:logs/failed-ops.log \  
  --set include-request-details-in-result-messages:true \  
  --set "rotation-policy:7 Days Time Limit Rotation Policy" \  
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. View the `failed-ops.log` in the `logs` directory. Verify that only information about failed operations is written to it.

Managing Admin Alert Access Logs

Admin Alert Access Logs are a specialized form of filtered log that automatically generates an administrative alert when criteria configured for the log publisher matches those messages in the access log.

About Access Log Criteria

Configuring an Admin Alert Access Log requires that you configure the criteria for the access log messages. Each criteria can be either a Simple or an Aggregate type. The Simple type uses the set of properties for the client connection, operation request, and the contents of

any operation-specific requests or results. The Aggregate type provides criteria that contains Boolean combination of other operation-specific criteria objects. For more information, see the *PingData Directory Server Configuration Reference*.

The criteria can be one or more of the following:

- **Connection Criteria.** Defines sets of criteria for grouping and describing client connections based on a number of properties, including protocol, client address, connection security, and authentication state.
- **Request Criteria.** Defines sets of criteria for grouping and describing operation requests based on a number of properties, including properties for the associated client connection, the type of operation, targeted entry, request controls, target attributes, and other operation-specific terms.
- **Result Criteria.** Defines sets of criteria for grouping and describing operation results based on a number of properties, including the associated client connection and operation request, the result code, response controls, privileges missing or used, and other operation-specific terms.
- **Search Entry Criteria.** Defines sets of criteria for grouping and describing search result entries based on a number of properties, including the associated client connection and operation request, the entry location and contents, and included controls.
- **Search Reference Criteria.** Defines sets of criteria for grouping and describing search result references based on a number of properties, including the associated client connection and operation request, reference contents, and included controls.

Configuring an Admin Alert Access Log Publisher

Prior to configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system. For more information, see [Working with Administrative Alert Handlers](#).

To Configure an Admin Alert Access Log Publisher

1. Use `dsconfig` to create a criteria object for the Admin Alert Access Log. For this example, we want to log only write operations that target user entries. The following command matches any of the specified operations whose target entry matches the filter `"(objectClass=person)"`.

If you are using the `dsconfig` tool in interactive mode, the menu items for the criteria operations are located in the Standard objects menu.

```
$ bin/dsconfig create-request-criteria --type simple \
  --criteria-name "User Updates" \
  --set operation-type:add \
  --set operation-type:delete \
  --set operation-type:modify \
  --set operation-type:modify-dn \
  --set "any-included-target-entry-filter:(objectClass=person)"
```

2. Use `dsconfig` to create a log publisher of type `admin-alert-access`.

```
$ bin/dsconfig create-log-publisher \
```

```
--publisher-name "User Updates Admin Alert Access Log" \  
--type admin-alert-access \  
--set "request-criteria:User Updates" \  
--set include-request-details-in-result-messages:true \  
--set enabled:true
```

Managing Syslog-Based Access Log Publishers

The Directory Server supports access logging using the syslog protocol that has been part of the Berkeley Software Distribution (BSD) operating systems for many years. Syslog provides a flexible, albeit simple, means to generate, store and transfer log messages that is supported on most UN*X and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 kbytes and has three important parts:

- **PRI.** Specifies the message priority based on its facility and severity. The message facility is a numeric identifier that specifies the type of log messages, such as kernel messages, mail system messages, etc. The severity is a numeric identifier that specifies the severity level of the operation that is being reported. The following values are for Solaris systems. For Linux systems, refer to the `syslog.h` file for the specific numbers on their target operating system:
 - > 0 (emergency level)
 - > 1 (Alert: action must be taken immediately)
 - > 2 (Critical: critical conditions)
 - > 3 (Error: error conditions)
 - > 4 (Warning: warning conditions)
 - > 5 (Notice: normal but significant condition)
 - > 6 (Informational: informational messages)
 - > 7 (Debug: debug level messages)

Together, the facility and the severity determine the priority of the log message indicated by angled brackets and 1-3 digit priority number. For example, "<0>", "<13>", "<103>" are valid representations of the PRI.

- **Timestamp and Host Name.** The timestamp displays the current date and time of the log. The host name or IP address displays the source of the log.
- **Message.** Displays the actual log message.

Administrators can configure syslog to handle log messages using log priorities that are based on the message's facility and severity. This feature allows users to configure the logging system in such a way that messages with high severities can be sent to a centralized repository, while lower severity messages can be stored locally on a server.



Note: Since the numeric values of the severity and facility are operating system-dependent, the central repository must only include syslog messages from compatible OS types, otherwise the meanings of the PRI field is ambiguous.

Before You Begin

You will need to identify the host name and port to which you want to connect. Because the Syslog Protocol uses User Datagram Protocol (UDP) packets, we highly recommend that you use `localhost` and utilize some additional logging tools, such as `syslog-ng`. UDP is unreliable and unsecure means to transfer data packets between hosts.

Default Access Log Severity Level

All messages are logged at the syslog severity level of 6, which is `Informational`: `informational` messages. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system.

Syslog Facility Properties

When using syslog, you have to specify a facility for the access log messages. As an advanced property, you can select a number that corresponds to the facility you wish to use. The default value for the `syslog-facility` property is 1 (one) for user-level messages. The following facility properties are available on the Directory Server. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system documentation.

Table 60: Syslog Facility Properties (Solaris)

Facility	Description
0	kernel messages
1	user-level messages (default)
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0
17	local use 1

Facility	Description
18	local use 2
19	local use 3
20	local use 4
21	local use 5
22	local use 6
23	local use 7

Queue-Size Property

The maximum number of log records that can be stored in the asynchronous queue is determined by the `queue-size` property. The default queue size set is 10000, which means that the server will continuously flush messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Therefore, lowering this value can impact performance.

Configuring a Syslog-Based Access Log Publisher

You can configure a Syslog-based Access Log Publisher using the `dsconfig` tool. We recommend that you use `syslog` locally on `localhost` and use `syslog-ng` to send the `syslog` messages to remote `syslog` servers.

Because `syslog` implementations differ by vendor, please review your particular vendor's `syslog` configuration.

To Configure a Syslog-Based Access Log Publisher

- Use `dsconfig` to create a log publisher of type `syslog-based-access`.
If you are using the `dsconfig` tool in interactive mode, the menu item for Syslog Facility is an advanced property, which can be exposed by typing `a` (for "show advanced properties") on the Syslog-Based Access Log Publisher menu.

```
$ bin/dsconfig create-log-publisher \  
--publisher-name "syslog-access" \  
--type syslog-based-access \  
--set syslog-facility:4 \  
--set enabled:true
```

Managing the File-Based Audit Log Publishers

The Directory Server provides an audit log, a specialized version of the access log, for troubleshooting problems that may occur in the course of processing. The log records all changes to the data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

Audit Log Format

The audit log uses standard LDIF format, so that administrators can quickly analyze what changes occurred to the data. The audit log begins logging when enabled and should be used to debug any issues that may have occurred. Some common properties are the following:

- **Timestamp.** Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>
- **Connection ID.** Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation.
- **Operation ID.** Numeric identifier, starting incrementally with 0, that identifies the operation.
- **Modifiers Name.** Displays the DN of the user who made the change.
- **Update Time.** Records the `modifyTimestamp` operational attribute.

Audit Log Example

The following example shows output from the Audit Log in the `<server-root>/ logs/audit`. The first entry shows when the audit log was enabled. The second entry show changes made to a user entry.

```
# 05/Jun/2011:10:29:04 -0500; conn=0; op=55
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z

# 05/Jun/2011:10:31:20 -0500; conn=2; op=1
dn: uid=user.996,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 115 426 4748
-
replace: homePhone
homePhone: +1 407 383 4949
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
```

Enabling the File-Based Audit Log Publisher

You can enable the File-Based Audit Log Publisher using the `dsconfig` tool. The audit log can impact the Directory Server's write performance, so enable it only when troubleshooting any issues.

To Enable the File-Based Audit Log Publisher

- Use `dsconfig` to enable the File-Based Audit Log Publisher. For this example, the instance name and startup ID is also enabled in the audit log.

```
$ bin/dsconfig set-log-publisher-prop \  
  --publisher-name "File-Based Audit Logger" \  
  --set enabled:true \  
  --set include-instance-name:true \  
  --set include-startup-id:true
```

Obscuring Values in the Audit Log

You can obscure the values of specific attributes in the audit log using the `obscure-attribute` property. Each value of an obscured attribute is replaced in the audit log with a string of the form "***** OBSCURED VALUE *****". By default, attributes are not obscured, because the values of password attributes appear in hashed form rather than in the clear.

Managing the JDBC Access Log Publishers

The Directory Server supports the Java Database Connectivity (JDBC™) API, which allows access to SQL data stores by means of its JDBC drivers. The JDBC 4.0 API, part of the Java SDK, provides a seamless method to interface with various database types in heterogeneous environments.

By easily connecting to a database, the Directory Server can be configured to implement a centralized logging system with different databases. Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. Some disadvantages of centralized logging are that data flow asymmetries may complicate synchronization or network provisioning and may unduly burden the central repository with possibly heavy loads.

Before You Begin

Before configuring the JDBC Access Log Publisher, you need to carry out two essential steps to set up the database.

- Install the database drivers in the Directory Server `lib` directory.

- Define the log mapping tables needed to map access log elements to the database column data. Only those elements in the log mapping table gets logged by the JDBC Log Publisher.

The following sections provide more information about these tasks.

Configuring the JDBC Drivers

The Directory Server supports a number of JDBC drivers available in the market. We highly recommend using the JDBC 4 drivers supported in the Java platform. For example, for Oracle databases, you need to use the `ojdbc.jar` driver for Java and any associated JAR files (National Language Support jars, and others) required to connect with the particular database. The following databases are supported:

- > DB2
- > MySQL
- > Oracle Call Interface (OCI)
- > Oracle Thin
- > PostgreSQL
- > SQL Server

To Configure the JDBC Driver

- Obtain the JAR file(s) for your particular database, and copy it into the `<server-root>/lib` directory.

Configuring the Log Field Mapping Tables

The log field mapping table associates access log fields to the database column names. You can configure the log field mapping table using the `dsconfig` tool, which then generates a DDL file that you can import into your database. The DDL file is generated when you create the JDBC Log Publisher.

To uniquely identify a log record, we recommend always mapping the following fields: `timestamp`, `startupid`, `message-type`, `connection-id`, `operation-type`, `instance-name`.

The table name is not part of this mapping.

The Directory Server also provides three options that you can quickly select for creating a log field mapping table:

- **Complete JDBC Access Log Field Mappings.** Maps all 52 object properties.
- **Complete JDBC Error Log Field Mappings.** Maps all 8 object properties.
- **Simple JDBC Access Log Field Mappings.** Maps a common set of object properties.
- **Custom JDBC Access Log Field Mappings.** Create a custom set of JDBC log field mappings.

- **Custom JDBC Error Log Field Mappings.** Create a custom set of JDBC error log field mappings.

To Configure the Log Field Mapping Tables

1. Use `dsconfig` to create a log field mapping table. On the main menu, type `o` to change to the Standard Object menu, and type the number corresponding to Log Field Mapping.
2. On the **Log Field Mapping management** menu, enter the option to create a new Log Field Mapping.
3. On the **Log Field Mapping template** menu, enter the option to select a complete JDBC Access Log Field mapping to use as a template for your new field mapping.
4. Next, enter a name for the new field mapping. In this example, type `my-jdbc-test`.
5. On the **Access Log Field Mapping Properties** menu, select a property for which you want to change the value. Any property that is undefined will not be logged by the JDBC Access Log Publisher. When complete, type `f` to save and apply the changes.
6. On the **Log Field Mapping Management** menu, type `q` to exit the menu.
7. View the existing Log Mappings on the system.

```
$ bin/dsconfig list-log-field-mappings
```

```
Log Field Mapping           : Type
-----
Complete JDBC Access Log Field Mappings : access
Complete JDBC Error Log Field Mappings  : error
my-jdbc-test                 : access
Simple JDBC Access Log Field Mappings    : access
```

Configuring the JDBC Access Log Publisher using dsconfig Interactive Mode

After setting up the drivers and the log mapping table, use the `dsconfig` utility to configure the JDBC Access Log Publisher on the Directory Server. The following example uses `dsconfig` interactive mode to illustrate the steps required to configure the log publisher and the external database server.

To Configure the JDBC Access Log Publisher

1. Copy the database JAR files to the `<server-root>/lib` directory, and then restart the Directory Server.
2. Launch the `dsconfig` tool in interactive command-line mode.

```
$ bin/dsconfig
```

3. Next, type the connection parameters to bind to the Directory Server. Enter the host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on

the Directory Server, the LDAP listener port number, the user bind DN, and the bind DN password.

4. On the main menu, type the number corresponding to Log Publisher.
5. On the **Log Publisher management** menu, enter the option to create a new Log Publisher.
6. On the **Log Publisher template** menu, type n to create a new Log Publisher.
7. On the **Log Publisher Type** menu, enter the option to create a new JDBC-Based Access Log Publisher.
8. Type a name for the JDBC Access Log Publisher.
9. On the **Enabled Property** menu, enter the option to enable the log publisher.
10. On the **Server Property** menu, enter the option to create a new JDBC External Server.
11. Next, type the name for the JDBC External Server. This is a symbolic name used to represent the DBMS.
12. On the **JDBC Driver Type Property** menu, type the number corresponding to the type of JDBC database driver type.
13. Next, type a name for the `database-name` property. This is the DBMS database name. The database name must contain the table referred to in the generated DDL.
14. Next, type the host name or IP address (`server-host-name`) of the external server.
15. Type the server listener port. In this example, type 1541.
16. Review the properties for the external server, and the type f to apply the changes.
17. If you need to supply your own JDBC URL, type a for advanced properties to open the `jdbc-driver-url` property and supply the appropriate URL. The example below shows how to access an Oracle Thin Client connection using a SID instead of a Service.

```
>>>> Configure the properties of the JDBC External Server

Property Value(s)
-----
1) description -
2) jdbc-driver-type oraclethin
3) jdbc-driver-url jdbc:oracle:thin@myhost:1541:my_SID
4) database-name jdbc-test
5) server-host-name localhost
6) server-port 1541
7) user-name -
8) password -

?) help
f) finish - create the new JDBC External Server
a) hide advanced properties of the JDBC External Server
d) display the equivalent dsconfig arguments to create this object
b) back
q) quit

Enter choice [b]: f

JDBC External Server was created successfully
```

18. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

Configuring the JDBC Access Log Publisher Using dsconfig Non-Interactive Mode

The following example uses `dsconfig` non-interactive mode to illustrate the steps to configure the log publisher and the external database server presented in the previous section.

To Configure the JDBC Access Log Publisher in Non-Interactive Mode

1. Use `dsconfig` with the `--no-prompt` option to create the JDBC external server.

```
$ bin/dsconfig --no-prompt create-external-server \
--server-name jdbc-external \ --type jdbc \
--set jdbc-driver-type:oraclethin \
--set database-name:ubid_access_log \
--set server-host-name:localhost --set server-port:1541
```

2. Use `dsconfig` to create the log publisher.

```
$ bin/dsconfig --no-prompt create-log-publisher \
--publisher-name jdbc-test \
--type jdbc-based-access \
--set enabled:true \
--set server:jdbc-external \
--set "log-field-mapping:Simple JDBC Access Log Field Mappings"
```

3. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

The procedure to configure the JDBC-Based Error Log Publisher is similar to creating a JDBC-Based Access Log Publisher. You can run the previous `dsconfig` command with the `--type jdbc-based-error` as follows:

```
$ bin/dsconfig --no-prompt create-log-publisher \
--publisher-name jdbc-error-test \
--type jdbc-based-error \
--set enabled:true \
--set server:jdbc-external \
--set "log-field-mapping:Simple JDBC Access Log Field Mappings"
```

Managing the File-Based Error Log Publisher

The Error Log reports errors, warnings, and informational messages about events that occur during the course of the Directory Server's operation. Each entry in the error log records the following properties (some are disabled by default and must be enabled):

- **Time Stamp.** Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>
- **Category.** Specifies the message category that is loosely based on the server components.

- **Severity.** Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are: fatal-error, notice, severe-error, severe-warning.
- **Message ID.** Specifies the numeric identifier of the message.
- **Message.** Stores the error, warning, or informational message.

Error Log Example

The following example displays the error log for the Data Metrics Server. The log is enabled by default and is accessible in the <server-root>/logs/errors file.

```
[21/Oct/2012:05:15:23.048 -0500] category=RUNTIME_INFORMATION severity=NOTICE
msgID=20381715 msg="JVM Arguments: '-Xmx8g', '-Xms8g', '-XX:MaxNewSize=1g',
'-XX:NewSize=1g', '-XX:+UseConcMarkSweepGC', '-XX:+CMSConcurrentMTEnabled',
'-XX:+CMSParallelRemarkEnabled', '-XX:+CMSParallelSurvivorRemarkEnabled',
'-XX:+CMSScavengeBeforeRemark', '-XX:RefDiscoveryPolicy=1',
'-XX:ParallelCMSThreads=4', '-XX:CMSMaxAbortablePreCleanTime=3600000',
'-XX:CMSInitiatingOccupancyFraction=80', '-XX:+UseParNewGC', '-XX:+UseMembar',
'-XX:+UseBiasedLocking', '-XX:+UseLargePages', '-XX:+UseCompressedOops',
'-XX:PermSize=128M', '-XX:+HeapDumpOnOutOfMemoryError',
'-Dcom.unboundid.directory.server.scriptName=setup'"
[21/Oct/2012:05:15:23.081 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555611 msg="Administrative alert type=server-starting
id=4178daee-ba3a-4be5-8e07-5ba17bf30b71
class=com.unboundid.directory.server.core.MetricsEngine
msg='The Data Metrics Server is starting'"
[21/Oct/2012:05:15:23.585 -0500] category=CORE severity=NOTICE
msgID=1879507338 msg="Starting group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=CORE severity=NOTICE
msgID=1879507339 msg="Completed group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555575 msg="'Group cache (2 static group(s) with 0 total
memberships and 0 unique members, 0 virtual static group(s),
1 dynamic group(s))' currently consumes 7968 bytes and can grow to a maximum
of an unknown number of bytes"
[21/Oct/2012:05:16:18.011 -0500] category=CORE severity=NOTICE
msgID=458887 msg="The Data Metrics Server (Data Metrics Server 6.0.1.0
build 20121021003738Z, R12799) has started successfully"
```

To Modify the File-Based Error Logs

- Use dsconfig to modify the default File-Based Error Log.

```
$ bin/dsconfig set-log-publisher-prop \
--publisher-name "File-Based Error Logger" \
--set include-product-name:true --set include-instance-name:true \
--set include-startup-id:true
```

Managing the Syslog-Based Error Log Publisher

The Directory Server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher. You can easily configure the error logger using the dsconfig tool.

Syslog Error Mapping

The Directory Server automatically maps error log severities to the syslog severities. The following mappings are used:

Table 61: Error to Syslog Severities Mappings to Syslog

Error Log Facility	Syslog Severity
FATAL_ERROR,0	Syslog Emergency
SEVERE_ERROR,1	Syslog Alert
SEVERE_WARNING,2	Syslog Critical
MILD_ERROR,3	Syslog Error
MILD_WARNING,4	Syslog Warn
NOTICE,5	Syslog Notice
INFORMATION,6	Syslog Info
DEBUG,7	Syslog Debug

Configuring a Syslog-Based Error Log Publisher

You can configure a Syslog-based Error Log Publisher using the `dsconfig` tool. Again, we recommend that you use `syslog` locally on `localhost` and use `syslog-ng` to send the data packets over the UDP protocol.

Because `syslog` implementations differ by vendor, please review your particular vendor's `syslog` configuration.

To Configure a Syslog-Based Error Log Publisher

- Use `dsconfig` to create a log publisher of type `syslog-based-error`. In this example, set the `syslog facility` to 4 for security/authorization messages.

```
$ bin/dsconfig create-log-publisher --publisher-name "syslog-error" \  
--type syslog-based-error --set syslog-facility:4 --set enabled:true
```

Creating File-Based Debug Log Publishers

The Directory Server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that might occur during server processing. Because the debug data may be too large to maintain during normal operations, the Debug Log Publisher must be specifically configured and enabled. The Debug Log reports the following types of information:

- Exception data thrown by the server.
- Data read or written to network clients.
- Data read or written to the database.

- Access control or password policy data made within the server.

You can use the `dsconfig` tool to create a debug log publisher. You should only create a debug logger when troubleshooting a problem due to the voluminous output the Directory Server generates.

To Create a File-Based Debug Log Publisher

- Use `dsconfig` to create the debug log publisher. The `log-file` property (required) sets the debug log path. You must also specify the rotation and retention policy for the debug log.

```
$ bin/dsconfig create-log-publisher \  
  --publisher-name "File-Based Debug Logger" \  
  --type file-based-debug \  
  --set enabled:true \  
  --set log-file:/logs/debug \  
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \  
  --set "rotation-policy:Size Limit Rotation Policy" \  
  --set "retention-policy:File Count Retention Policy"
```

To Delete a File-Based Debug Log Publisher

- Use `dsconfig` to delete the debug log publisher.

```
$ bin/dsconfig delete-log-publisher \  
  --publisher-name "File-Based Debug Logger"
```

Chapter

21

Managing Monitoring

The Ping Identity Directory Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the PingData® Data Metrics Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

Topics:

- [The Monitor Backend](#)
- [Monitoring Disk Space Usage](#)
- [Monitoring with the Data Metrics Server](#)
- [About the Collection of System Monitoring Data](#)
- [Monitoring Key Performance Indicators by Application](#)
- [Configuring the External Servers](#)
- [Preparing the Servers Monitored by the Data Metrics Server](#)
- [Configuring the Processing Time Histogram Plugin](#)
- [Setting the Connection Criteria to Collect SLA Statistics by Application](#)
- [Updating the Global Configuration](#)
- [Proxy Considerations for Tracked Applications](#)
- [Monitoring Using SNMP](#)
- [SNMP Implementation](#)
- [Configuring SNMP](#)
- [Configuring SNMP on AIX](#)
- [MIBS](#)
- [Monitoring with the Administrative Console](#)
- [Accessing the Processing Time Histogram](#)
- [Monitoring with JMX](#)
- [Running JConsole](#)
- [Monitoring the Directory Server Using JConsole](#)
- [Monitoring Using the LDAP SDK](#)
- [Monitoring over LDAP](#)
- [Profiling Server Performance Using the Stats Logger](#)
- [Adding Custom Logged Statistics to a Periodic Stats Logger](#)

The Monitor Backend

The Directory Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingData Data Metrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the `bin/status` tool to display server component activity and state.

The list of all monitor entries can be seen using `ldapsearch` as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
--baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Table 62: Directory Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., <code>connID="15"</code> , <code>connectTime="20100308223038Z"</code> , etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Server.
General	Provides general information about the state of the Directory Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for <code>read/write/remove/open-cursor/read-for-search</code> . These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.

Component	Description
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Server version, including build ID, version, revision number, etc.
Work Queue	<p>Provides information about the state of the Directory Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a <code>monitor-queue-time</code> property set to <code>true</code> by default. This logs messages for new operations with a <code>qtime</code> attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue.</p> <p>A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the <code>ldapsearch</code> command for example, use the <code>--useAdministrativeSession</code> option. The requester must have the <code>use-admin-session</code> privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the <code>num-administrative-session-worker-threads</code> property in the work queue configuration.</p>

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the `DiskSpaceConsumer` interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- Low space warning threshold.** This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time

the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.

- **Low space error threshold.** This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.
- **Out of space error threshold.** This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the Ping Identity Directory Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the Data Metrics Server

The Data Metrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various PingData servers in a deployment. The Data Metrics Server relies on a captive PostgreSQL data store for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your PingData Platform system. For more information, see the *PingData Data Metrics Server Administration Guide*.

About the Collection of System Monitoring Data

All PingData servers have the capability to monitor the health of the server and host system they run on for diagnostic review and troubleshooting. Initially, the servers do not collect any performance data until they are prepared for monitoring by a PingData Data Metrics Server using the `monitored-servers add-servers` tool or an administrator enables system health data collection for real-time inspection and querying. At a high level, all of the important server and machine metrics which can be monitored are available in the `cn=monitor` backend.

The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and what kind of data aggregation occurs for LDAP application statistics. The Stats Collector plugin ensures that a Data Metrics Server is able to gather all of the detailed data required for a comprehensive diagnostic review.

The Stats Collector plugin relies exclusively on entries in the `cn=monitor` backend to sample data using LDAP queries. The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and the type of data aggregation that occurs for LDAP application statistics. The Stats Collector plugin is configured with the `dsconfig` tool and collects data using LDAP queries. For example, the `--server-info:extended` option includes collection for the following:

- CPU
- JVM memory
- Memory
- Disk information
- Network information

Utilization metrics are gathered via externally invoked OS commands, such as `iostat` and `netstat`, using platform-specific arguments and version-specific output parsing.

Enabling the Host System monitor provider automatically gathers CPU and memory utilization but only optionally gathers disk and network information. Disk and network interfaces are enumerated in the configuration by device names (e.g., `eth0` or `lo`), and by disk device names (e.g., `sdl`, `sdab`, `sda2`, `scsi0`).

Monitoring Key Performance Indicators by Application

The Ping Identity Directory Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the PingData identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the PingData Data Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Server Configuration section of the *PingData Data Metrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the Data Metrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the Data Metrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the External Servers

Before you install the Data Metrics Server, you need to configure the servers you will be monitoring: Ping Identity Directory Server, Ping Identity Directory Proxy Server, and Data Sync Server. The Data Metrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Server, you can use the `dsconfig` tool to make configuration changes for the Data Metrics Server. When using the `dsconfig` tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the Servers Monitored by the Data Metrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Server keeps a metric history so that the Data Metrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the Data Metrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the Data Metrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the Data Metrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram Plugin

The Processing Time Histogram plugin is configured on each Directory Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the Data Metrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the Data Metrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the `separate-monitor-entry-per-tracked-application` property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the `separate-monitor-entry-per-tracked-application` property is set to true, then the `per-application-ldap-stats` property must be set to `per-application-only` on the Stats Collector Plugin and vice versa.

For example, the following `dsconfig` command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Processing Time Histogram" \
  --set separate-monitor-entry-per-tracked-application:true
```

The following `dsconfig` command line sets the `per-application-ldap-stats` property of the Stats Collector plugin to `per-application-only`:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \  
--set per-application-ldap-stats:per-application-only
```

Setting the Connection Criteria to Collect SLA Statistics by Application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the `application-name` dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the `included-user-base-dn` property to include the Directory Manager's full LDIF entry.

The following `dsconfig` command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \  
--criteria-name "Directory Manager" \  
--type simple \  
--set "included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config"
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following `dsconfig` command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \  
--set "tracked-application:Directory Manager"
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy Considerations for Tracked Applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the `tracked-application` property.

Monitoring Using SNMP

The Ping Identity Directory Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The Ping Identity Directory Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Server's SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

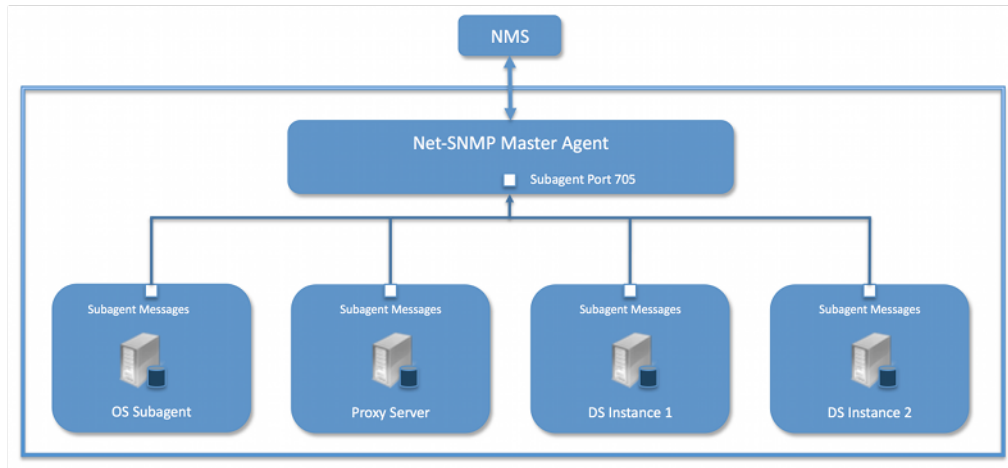


Figure 20: Example SNMP Deployment

One important note is that the Ping Identity Directory Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module, such as the System Management Agent (SMA) on Solaris or OpenSolaris. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.



Note: The Directory Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Server instance, enable

the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, `public` (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (`-f`), do not read any configuration files (`-C`) except the one specified with the `-c` option, print to standard output (`-Lo`), and then specify that debugging output should be turned on for the User-based Security Module (`-Dusm`). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`).

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
-m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: `-v <SNMP version>`, `-u <user name>`, `-A <user password>`, `-l <security level>`, `-n <context name (instance name)>`. The `-m all` option loads all MIBs in the default MIB directory in `/usr/share/snmp/mibs` so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost systemStatus
```

Configuring SNMP on AIX

Native AIX SNMP implementations do not support AgentX sub-agents, which is a requirement for the Ping Identity Directory Server. To implement SNMP on AIX platforms, any freely-available `net-snmp` package must be installed.

Special care must be made to ensure that you are using the `net-snmp` binary packages and not the native `snmp` implementation. Third-party `net-snmp` binary packages typically install under `/opt/freeware` and have the following differences:

```
Native Daemon: /usr/sbin/snmpd
Native Configuration File: /etc/snmpd.conf, /etc/snmpdv3.conf
Native Daemon Start and Stop: startsrc -s snmpd, stopsrc -s snmpd

net-snmp Daemon: /opt/freeware/sbin/snmpd
net-snmp Configuration File: /opt/freeware/etc/snmp/snmpd.conf
net-snmp start and stop: /etc/rc.d/init.d/snmpd start|stop
```

When configuring an SNMP implementation on AIX, remember to check the following items so that the Directory Server is referencing the `net-snmp` installation:

- The shell `PATH` will reference the native implementation binaries. Adjust the `PATH` variable or invoke the `net-snmp` binaries explicitly.
- If the native daemon is not stopped, there will likely be port conflicts between the native daemon and the `net-snmp` daemon. Disable the native daemon or use distinct port numbers for each.

SNMP on AIX Security Considerations

On AgentX sub-agent-compliant systems, it is recommended to use `agentXSocket tcp:localhost:705` to configure the net-snmp master agent to allow connections only from sub-agents located on the same host. On AIX systems, it is possible to specify an external IP network interface (for example, `agentXSocket tcp:0.0.0.0:708` would listen on all external IP interfaces), which would allow the Ping Identity Directory Server to be located on a different host to the snmp master agent.

While it is possible to implement non-local sub-agents, administrators should understand the security risks that are involved with this configuration. Primarily, because there is no communication authentication or privacy between the Ping Identity Directory Server and the master agent. An eavesdropper might be able to listen in on the monitoring data sent by the Ping Identity Directory Server. Likewise, a rogue sub-agent might be able to connect to the master agent and provide false monitoring data or deny access to SNMP monitoring data.

In general, it is recommended that sub-agents be located on the same host as the master agent.

MIBS

The Directory Server provides SMIV2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB.** Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB.** Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB.** Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB.** Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB.** Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB.** Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

PingData has a Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

The Ping Identity Directory Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Administrative Console, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The Ping Identity Directory Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the

monitor information that the Directory Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Server using the `dsconfig` tool. See [Configuring the JMX Connection Handler and Alert Handler](#).

To invoke the JConsole executable, type `jconsole` on the command line. If `JDK_HOME` is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To Run JConsole

1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

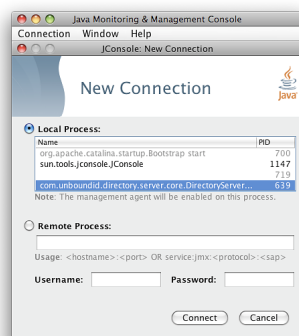
```
$ jconsole
```

Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

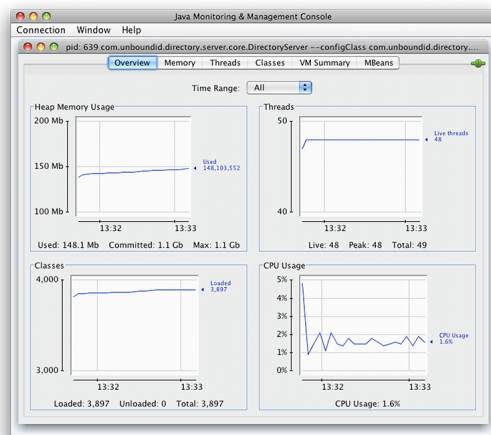


```
$ jconsole \
-J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
-J-Djavax.net.ssl.trustStorePassword=secret \
-J-Djava.class.path=$SERVER_ROOT/PingDirectory.jar:/Library/Java/
JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/jconsole.jar
```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



3. Review the resource monitoring information.



Monitoring the Directory Server Using JConsole

You can set up JConsole to monitor the Directory Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Server using JConsole

1. Start the Directory Server.

```
$ bin/start-ds
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

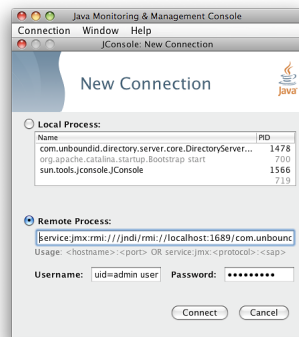
3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

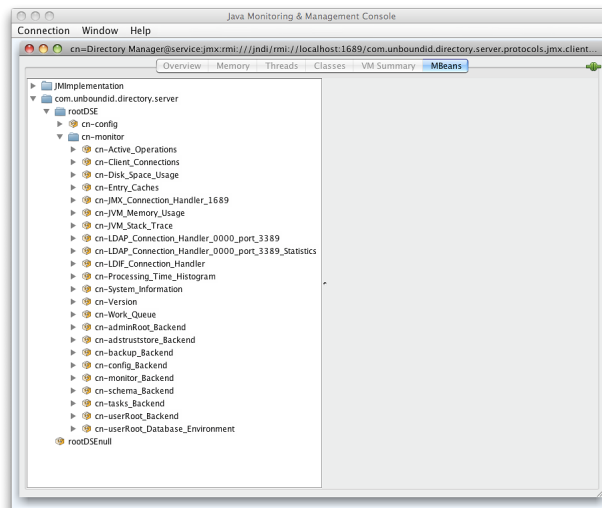
4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

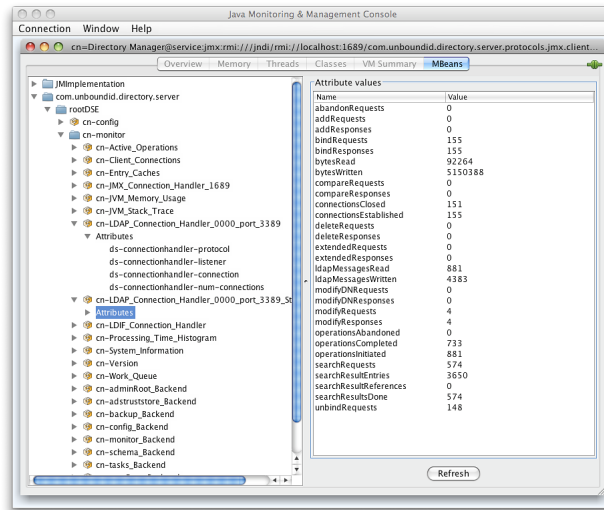
- In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



- Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



- Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
{
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
        for (Object value : a.getValues())
        {
            System.out.println(" " + a.getDisplayName() + ": " + String.valueOf(value));
        }
    }
    System.out.println();
}
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The Ping Identity Directory Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
--bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
--baseDN "cn=monitor" "(objectclass=*)"
```


Profiling Server Performance Using the Stats Logger

The Directory Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

To Enable the Stats Logger

By default, the Directory Server ships with the built-in 'Stats Logger' disabled. To enable it using the `dsconfig` tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select Stats Logger.

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plug-in.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin
```

Property	Value(s)
1) description	Logs performance stats to a log file periodically.
2) enabled	false
3) local-db-backend-info	basic
4) replication-info	basic
5) entry-cache-info	-
6) host-info	-
7) included-ldap-application	If per-application LDAP stats is enabled, then stats will be included for all

```

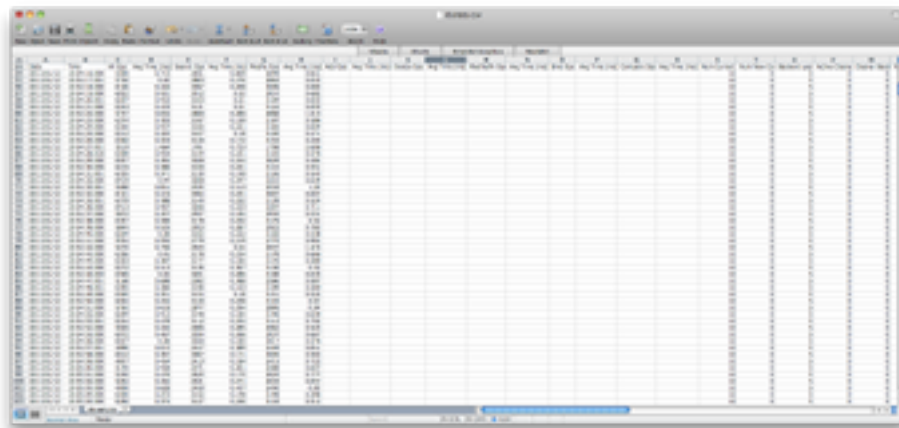
8) log-interval applications.
1 s
9) collection-interval 200 ms
10) suppress-if-idle true
11) header-prefix-per-column false
12) empty-instead-of-zero true
13) lines-between-header 50
14) included-ldap-stat active-operations, num-connections,
op-count-and-latency, work-queue
15) included-resource-stat memory-utilization
16) histogram-format count
17) histogram-op-type all
18) per-application-ldap-stats aggregate-only
19) ldap-changelog-info -
20) gauge-info none
21) log-file logs/dsstats.csv
22) log-file-permissions 640
23) append true
24) rotation-policy Fixed Time Rotation Policy, Size Limit
Rotation Policy
25) retention-policy File Count Retention Policy

?) help
f) finish - apply any changes to the Periodic Stats Logger Plugin
a) hide advanced properties of the Periodic Stats Logger Plugin
d) display the equivalent dsconfig command lines to either re-create this
object or only to apply pending changes
b) back
q) quit

Enter choice [b]:

```

7. Run the Directory Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate --help` to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plug-in.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plug-in as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `StatsLogger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to `10s`, and the `suppress-if-idle` to `false`. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.



Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to `memory-utilization` on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the `objectclass` attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter `1073741824` (i.e., `1073741824` bytes = 1 gigabytes).

14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
>>>> via creating 'Memory Usage' Custom Logged Stats
```

Property	Value(s)
1) description	-
2) enabled	true
3) monitor-objectclass	ds-memory-usage-monitor-entry
4) include-filter	-
5) attribute-to-log	total-bytes-used-by-memory-consumers
6) column-name	Memory Consumer Total (GB)
7) statistic-type	raw
8) header-prefix	-
9) header-prefix-attribute	-
10) regex-pattern	-
11) regex-replacement	-
12) divide-value-by	1073741824
13) divide-value-by-attribute	-
14) decimal-format	#.##
15) non-zero-implies-not-idle	false

```
?) help
f) finish - create the new Custom Logged Stats
a) hide advanced properties of the Custom Logged Stats
d) display the equivalent dsconfig arguments to create this object
b) back
q) quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

To Configure a Custom Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
--stats-name "Memory Usage" --type custom \
--set monitor-objectclass:ds-memory-usage-monitor-entry \
--set attribute-to-log:total-bytes-used-by-memory-consumers \
--set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
--set divide-value-by:1073741824
```


Chapter

22 Managing Notifications and Alerts

The Ping Identity Directory Server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, JMX, or SNMP in addition to standard error logging. Alerts and events reflect state changes within the server that may be of interest to a user or monitoring service. Notifications are typically the delivery of an alert or event to a user or monitoring service. Account status notifications are only delivered to the account owner notifying a change in state in the account.

This chapter presents the following topics:

Topics:

- [*Working with Account Status Notifications*](#)
- [*Working with Administrative Alert Handlers*](#)
- [*Configuring the JMX Connection Handler and Alert Handler*](#)
- [*Configuring the SMTP Alert Handler*](#)
- [*Configuring the SNMP Subagent Alert Handler*](#)
- [*Working with the Alerts Backend*](#)
- [*Working with Alarms, Alerts, and Gauges*](#)
- [*Testing Alerts and Alarms*](#)

Working with Account Status Notifications

The Ping Identity Directory Server supports notification handlers that can be used to notify users and/or administrators of significant changes related to password policy state for user entries. The following two notification handlers are available:

- **Error Log Account Status Notification Handler.** Enabled by default. The handlers send alerts to the error log when an account event occurs.
- **SMTP Account Status Notification Handler.** Disabled by default. You can enable the SMTP Handler with the `dsconfig` command to send notifications to designated email addresses.

Account Status Notification Types

The handlers send alerts when one of the account status events described in the following table occurs during password policy processing.

Table 63: Account Status Notification Types

Account Status Notification Types	Description
account-disabled	Generates a notification whenever a user account is disabled by an administrator.
account-enabled	Generates a notification whenever a user account is enabled by an administrator.
account-expired	Generates a notification whenever a user authentication attempt fails because the account has expired.
account-idle-locked	Generates a notification whenever a user authentication attempt fails because the account has been locked after idling for too long.
account-permanently-locked	Generates a notification whenever a user account is permanently locked (requiring administrative action to unlock the account) after too many failed attempts.
account-reset-locked	Generates a notification whenever an authentication attempt fails because the user account is locked because the user failed to change a password within the required interval that was reset by an administrator.
account-temporarily-locked	Generates a notification whenever a user account is temporarily locked after too many failed attempts.
account-unlocked	Generates a notification whenever a user account is unlocked by an administrator.
password-changed	Generates a notification whenever a user changes his or her own password.
password-expired	Generates a notification whenever a user authentication fails because the password has expired.
password-expiring	Generates a notification the first time that a password expiration warning is encountered for a user password.
password-reset	Generates a notification whenever a user's password is reset by an administrator.

Working with the Error Log Account Status Notification Handler

The Error Log Account Status Notification Handler is enabled by default and sends alerts when one of the account status events occur.

To Disable the Error Log Account Status Notification Handler

- Use the `dsconfig` tool to disable the Error Log Handler. You can view the log at `logs/error`.

```
$ bin/dsconfig set-account-status-notification-handler-prop \  
  --handler-name "Error Log Handler" --set enabled:false
```

To Remove a Notification Type from the Error Log Handler

- While not recommended, if you want to remove an account status notification type, use the `dsconfig` tool with the `--remove` option.

```
$ bin/dsconfig set-account-status-notification-handler-prop \  
  --handler-name "Error Log Handler" \  
  --remove account-status-notification-type: password-reset
```

Working with the SMTP Account Status Notification Handler

You can enable account status notifications to be sent to designated email addresses of end users, administrators, or both through an outgoing SMTP server. The email message is automatically generated from template files that contain the text to use in the message body. For example, the message subject for the account-disabled event is:

```
account-disabled: Your directory account has been disabled.
```

The message templates are located in the `config/messages` directory. The typical message body template is as follows:

```
Your directory account has been disabled.  
For further assistance, please contact a server administrator.
```

By default, the sender address is `notifications@example.com`, but you can configure your own address.

Before you enable the SMTP Account Status Notification Handler, you must configure the Directory Server to use at least one mail server as shown below. You can configure an SMTP server using `dsconfig` and the `set-global-configuration-prop` option.

To Configure the SMTP Server

1. Use `dsconfig` to configure a simple mail server.

```
$ bin/dsconfig create-external-server --server-name smtp1 \  
--type smtp --set server-host-name:smtp.example.com
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \  
--set smtp-server:smtp1
```

To Configure a StartTLS Connection to the SMTP Server

1. Use `dsconfig` to configure a StartTLS connection to the server.

```
$ bin/dsconfig create-external-server \  
--server-name myTLSServer --type smtp \  
--set server-host-name:tls.smtp.example.com \  
--set server-port:587 \  
--set smtp-security:starttls \  
--set user-name:MyAccountName \  
--set "password:"password:AAD5yZ+DjvwiYkBSMer6GQ6B3szQ6gSSBjA="
```

2. Use `dsconfig` to configure a newly-created SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \  
-set smtp-server:myTLSServer
```

To Configure an SSL Connection to the SMTP Server

1. Use `dsconfig` to create an external SMTP server using SSL.

```
$ bin/dsconfig create-external-server \  
--server-name ssl.smtp.example.com \  
--type smtp --set server-host-name:smtp.gmail.com \  
--set server-port:465 \  
--set smtp-security:ssl \  
--set 'username:my.name@example.com' \  
--set password:xxxxxx --set "smtp-timeout:10s"
```

2. Use `dsconfig` to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \  
--set smtp-server:ssl.smtp.example.com
```

To Enable the SMTP Account Status Notification Handler

- Use `dsconfig` to enable the SMTP account status notification handler.

```
$ bin/dsconfig set-account-status-notification-handler-prop \  
--handler-name "SMTP Server" --set enabled:true \  
--set "recipient-address:admin@example.com" \  
--set "sender-address:acct-status-notifications@example.com"
```

To View the Account Status Notification Handlers

- After you have enabled the SMTP server, view the list of account status notification handlers using `dsconfig`.

```
$ bin/dsconfig list-account-status-notification-handlers

Account Status Notification Handler : Type      : enabled
-----:-----:-----
Error Log Handler                  : error-log : true
SMTP Handler                        : smtp      : true
```

Associating Account Status Notification Handlers with Password Policies

To generate notifications whenever appropriate password policy state changes occur in the server, the password policy that governs the entry being updated must be configured to use one or more account status notification handlers. By default, password policies are not configured with any such handlers, and therefore, no account status notifications will be generated.

The set of account status notification handlers that should be in use for a password policy is controlled by the `account-status-notification-handler` property for that password policy. It can be configured using `dsconfig` or the Administrative Console. For example, the following change updates the default password policy, so that the error log account status notification handler will be invoked for any appropriate password policy state changes for entries governed by the default password policy:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

Working with Administrative Alert Handlers

The Ping Identity Directory Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler.** Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler.** Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler.** Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.

- **JMX Alert Handler.** Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. PingData uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler.** Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler.** Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler.** Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler.** Provides alert handler implementations created in third-party code using the Server SDK.

Administrative Alert Types

If enabled, the Directory Server can generate administrative alerts when the events occur. A full listing of system alerts and their severity is available in `<server-root>/docs/admin-alerts-list.csv`

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
```

```
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
--set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

Configuring the SMTP Alert Handler

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
--handler-name "SMTP Alert Handler" \
--type smtp \
--set enabled:true \
--set "sender-address:alerts@example.com" \
--set "recipient-address:administrators@example.com" \
--set "message-subject:Directory Admin Alert \%%alert-type\%%" \
--set "message-body:Administrative alert:\n\%%alert-message\%%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the `dsconfig` tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see [Monitoring Using SNMP](#).

To Configure the SNMP Subagent Alert Handler

- Use `dsconfig` to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

The Directory Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
--handler-name "SNMP Subagent Alert Handler" \
--set enabled:true \
--set server-host-name:host2 \
--set server-port:162 \
--set community-name:public
```

Working with the Alerts Backend

The Directory Server stores recently generated admin alerts in an Alerts Backend under the `cn=alerts` branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote monitoring. The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries in the `cn=alerts` backend.

The alerts persist on disk in the `config/alerts.ldif` file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the `dsconfig` tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the status command-line tool and in the Administrative Console.

To View Information in the Alerts Backend

- The following uses `ldapsearch` to view the admin alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
--bindPassword secret --baseDN cn=alerts "(objectclass=ds-admin-alert)"

dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.directory.server
ds-alert-message: The Directory Server has started successfully
```

To Modify the Alert Retention Time

1. Use `dsconfig` to change the maximum time information about generated admin alerts is retained in the Alerts backend. After this time, the information gets purged from the Directory Server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
--set "alert-retention-time: 2 weeks"
```

2. View the property using `dsconfig`.

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time
```

```
Property : Value(s)
-----:-----
alert-retention-time : 2 w
```

To Configure Duplicate Alert Suppression

- Use `dsconfig` to configure the maximum number of times an alert is generated within a particular timeframe for the same condition. The `duplicate-alert-time-limit` property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework. The `duplicate-alert-limit` property specifies the maximum number of duplicate alert messages should be sent over the administrative alert framework in the time limit specified in the `duplicate-alert-time-limit` property.

```
$ bin/dsconfig set-global-configuration-prop \
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

Working with Alarms, Alerts, and Gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or 'server unavailable' (Indicator Gauge). Gauges generate alarms, when the gauge's severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To View Information in the Alarms Backend

- The following uses `ldapsearch` to view alarms. The following displays the listing for the CPU usage alarm.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alarms "(objectclass=ds-admin-alarm)"
```

```
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
objectClass: top
objectClass: ds-admin-alarm
ds-alarm-id: CPU Usage (Percent)-Host System
ds-alarm-condition: CPU Usage (Percent)
ds-alarm-specific-resource: Host System
ds-alarm-severity: CRITICAL
ds-alarm-previous-severity: CRITICAL
ds-alarm-details: Gauge CPU Usage (Percent) for Host System
                   has value 99, having had a value of 83.13 in the
                   previous interval. The severity is critical, having
                   assumed this severity Thu Sep 25 10:24:20 CDT 2014
                   when the value crossed threshold 80
ds-alarm-additional-text: If CPU use is high, check the server's current workload
                        and other processes on this system and make any needed adjustments. Reducing
                        the load on the system will lead to better response times
ds-alarm-start-time: 20140925152420.004Z
ds-alarm-critical-last-time: 20140925152420.004Z
ds-alarm-critical-total-duration-millis: 0
```

Testing Alerts and Alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

To Test Alarms and Alerts

1. Configure a gauge with `dsconfig` and set the `override-severity` property to `critical`. The following example uses the CPU Usage (Percent) gauge.


```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

- Run the `status` tool to verify that an alarm was generated with corresponding alerts. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status
```

```

          --- Administrative Alerts ---
Severity : Time           : Message
-----
Info     : 11/Aug/2014      : A configuration change has been made in the
          : 15:48:46 -0500      : Directory Server:
          :                       : [11/Aug/2014:15:48:46.054 -0500]
          :                       : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                       : DNs,cn=config' authtype=[Simple] from=127.0.0.1
          :                       : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                       : --gauge-name 'Cleaner Backlog (Number Of Files)'
          :                       : --set warning-value:-1'
Info     : 11/Aug/2014      : A configuration change has been made in the
          : 15:47:32 -0500      : Directory Server: [11/Aug/2014:15:47:32.547 -0500]
          :                       : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                       : DNs,cn=config' authtype=[Simple] from=127.0.0.1
          :                       : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                       : --gauge-name 'Cleaner Backlog (Number Of Files)'
          :                       : --set warning-value:0'
Error    : 11/Aug/2014      : Alarm [CPU Usage (Percent). Gauge CPU Usage (Percent)
          : 15:41:00 -0500      : for Host System has
          :                       : a current value of '18.583333333333332'.
          :                       : The severity is currently OVERRIDDEN in the
          :                       : Gauge's configuration to 'CRITICAL'.
          :                       : The actual severity is: The severity is
          :                       : currently 'NORMAL', having assumed this severity
          :                       : Mon Aug 11 15:41:00 CDT 2014. If CPU use is high,
          :                       : check the server's current workload and make any
          :                       : needed adjustments. Reducing the load on the system
          :                       : will lead to better response times.
          :                       : Resource='Host System']
          :                       : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48 hours
Use the --maxAlerts and/or --alertSeverity options to filter this list

```

```

          --- Alarms ---
Severity : Severity Start : Condition : Resource : Details
          : Time              :           :          :
-----
Critical : 11/Aug/2014        : CPU Usage : Host System : Gauge CPU Usage (Percent) for
          : 15:41:00 -0500    : (Percent) :             : Host System
          :                   :           :             : has a current value of
          :                   :           :             : '18.785714285714285'.
          :                   :           :             : The severity is currently
          :                   :           :             : 'CRITICAL', having assumed
          :                   :           :             : this severity Mon Aug 11
          :                   :           :             : 15:49:00 CDT 2014. If CPU use
          :                   :           :             : is high, check the server's
          :                   :           :             : current workload and make any
          :                   :           :             : needed adjustments. Reducing
          :                   :           :             : the load on the system will
          :                   :           :             : lead to better response times
Warning  : 11/Aug/2014        : Work Queue: Work Queue : Gauge Work Queue Size (Number
          : 15:39:40 -0500    : Size      :             : of Requests) for Work Queue
          :                   : (Number of :             : has a current value of '27'.
          :                   : Requests) :             : The severity is currently
          :                   :           :             : 'WARNING' having assumed this
          :                   :           :             : severity Mon Aug 11 15:48:50
          :                   :           :             : CDT 2014. If all worker
          :                   :           :             : threads are busy processing
          :                   :           :             : other client requests, then
          :                   :           :             : new requests that arrive will
          :                   :           :             : be forced to wait in the work
          :                   :           :             : queue until a worker thread
          :                   :           :             : becomes available

```

Shown are alarms of severity [Warning,Minor,Major,Critical]
 Use the --alarmSeverity option to filter this list

Indeterminate Alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the `status` tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the `status` tool run with the `--alarmSeverity=indeterminate` option:

```

--- Alarms ---
Severity      : Severity Start : Condition      : Resource      : Details
              : Time             :                :               :
-----:-----:-----:-----:-----
Normal       : 26/Aug/2014     : Startup Begun : cn=config     : The Directory Server
              : 14:16:29 -0500 :                :               : is starting.
              :                 :                :               :
Indeterminate: 26/Aug/2014     : Replication   : not          : The value of gauge
              : 14:16:40 -0500 : Latency       : available    : Replication Latency
              :                 : (Milliseconds) :              : (Milliseconds) could not
              :                 :                :              : be determined. The
              :                 :                :              : severity is INDETERMINATE,
              :                 :                :              : having assumed this
              :                 :                :              : severity Tue Aug 26
              :                 :                :              : 14:17:10 CDT 2014.
    
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge's entry. The result is an error message may explain the indeterminate severity:

```

# ldapsearch -w password --baseDN "cn=monitor" \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn: Gauge Replication Latency (Milliseconds)
gauge-name: Replication Latency (Milliseconds)
resource:
severity: indeterminate
summary: The value of gauge Replication Latency (Milliseconds) could not
         be determined. The severity is INDETERMINATE, having assumed
         this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
              class ds-replica-monitor-entry
...
    
```

Chapter

23 Managing the SCIM Servlet Extension

The Ping Identity Directory Server provides a System for Cross-domain Identity Management (SCIM) servlet extension to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Topics:

- [Overview of SCIM Fundamentals](#)
- [Configuring SCIM](#)
- [Configuring Advanced SCIM Extension Features](#)
- [Configuring the Identity Access API](#)
- [Monitoring the SCIM Servlet Extension](#)

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet the your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

Summary of SCIM Protocol Support

PingData implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported.

Table 64: SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingData extension to the basic SCIM functionality.

About the Identity Access API

The Ping Identity Directory Server, Ping Identity Directory Proxy Server, and Data Sync Server support an extension to the SCIM standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the Ping Identity Directory Server.

Configuring SCIM

This section discusses details about the PingData implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Creating Your Own SCIM Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM SDK is available for download at <https://github.com/unboundid/scim>.



Note: The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the `scim-resource.xml` config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring the SCIM Servlet Extension

The Directory Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a `dsconfig` batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See [Managing the SCIM Schema](#) for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

To Configure SCIM Manually

The following example procedure assumes that you have configured the Directory Server using the default settings, which means that SSL and the HTTPS Connection Handler have not been configured. The example also shows the `dsconfig` non-interactive commands. You can easily use the `dsconfig` interactive commands, which uses a menu-driven interface. If you use the `dsconfig` interactive, you must change to the Standard or Advanced object menu to access many of these configuration settings.

1. Set up your certificates. Follow the examples shown in the section *Managing Certificates*. You should have a keystore and truststore set up in the `config` directory. Make sure that the `keystore.pin` and `truststore.pin` are set.
2. Enable the key manager provider. The key manager provider accesses the certificate during the SSL handshaking process. If running `dsconfig` interactive, open the main menu, select "Key Manager Provider" -> "View and edit an existing Key Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-key-manager-provider --provider-name JKS \  
--type file-based --set enabled:true --set key-store-file:config/keystore \  
--set key-store-type:JKS --set key-store-pin-file:config/keystore.pin
```

3. Enable the trust manager provider. The trust manager provider determines if a presented certificate can be trusted. If running `dsconfig` interactive, open the main menu, select "Trust Manager Provider" -> "View and edit an existing Trust Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-trust-manager-provider --provider-name JKS \  
--type file-based --set enabled:true --set trust-store-file:config/truststore \  
--set trust-store-type:JKS --set trust-store-pin-file:config/truststore.pin
```

- Configure the HTTPS Connection Handler. The command enables the connection handler, specifies the SCIM HTTP servlet extension and sets the listen port to a port of your choice, in this example, use 8443. The command also specifies the type of key manager and trust manager providers and sets the log publisher to "HTTP Detailed Access." If running `dsconfig` interactive, open the main menu, select "Connection Handler" -> "View and edit an existing Connection Handler" -> "HTTPS Connection Handler". Change the parameters to match your setup, and then, click "finish" to save the setting.

```
$ bin/dsconfig create-connection-handler \
--handler-name "HTTPS Connection Handler" \
--type http --set enabled:true \
--set listen-port:8443 \
--set use-ssl:true \
--set http-servlet-extension:SCIM \
--set "http-operation-log-publisher:HTTP Detailed Access" \
--set key-manager-provider:JKS --set trust-manager-provider:JKS
```

- Add access controls to allow access to LDAP controls used by the SCIM Servlet Extension. These controls are the Post-Read Request Control (1.3.6.1.1.13.2), Server-Side Sort Request Control (1.2.840.113556.1.4.473), Simple Paged Results Control (1.2.840.113556.1.4.319), and Virtual List View Request Control (2.16.840.1.113730.3.4.9). We recommend using the following command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetcontrol="1.3.6.1.1.13.2 || 1.2.840.113556.1.4.473 ||
1.2.840.113556.1.4.319 || 2.16.840.1.113730.3.4.9")
(version 3.0;acl "Authenticated access to controls used by the SCIM servlet
extension"; allow (all) userdn="ldap:///all");'
```

- Add access controls to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its `dsconfig` interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id ||
createTimestamp || modifyTimestamp")
(version 3.0;acl "Authenticated read access to operational attributes \
used by the SCIM servlet extension"; allow (read,search,compare)
userdn="ldap:///all");'
```

- Optional. The SCIM HTTP Connection Handler automatically uses a detailed HTTP log publisher, which is implemented in a proprietary format. If you need a standard W3C common log format publisher, enter the following command. If running `dsconfig` interactive, open the main menu, select "Log Publisher" -> "Create a new Log Publisher" -> "new Log Publisher created from scratch" -> "File Based Access Log Publisher", enter the parameters to match your setup, and then, click "finish" to save the setting. Go back to the main menu, select "Connection Handler" -> "HTTPS Connection Handler", and then add "HTTP Common Access" to the `http-operation-log-publisher` property. Click "finish" to save the setting.

```
$ bin/dsconfig create-log-publisher \
--publisher-name "HTTP Common Access" \
--type common-log-file-http-operation --set enabled:true \
--set log-file:logs/http-common-access \
--set "rotation-policy:24 Hours Time Limit Rotation Policy" \
--set "rotation-policy:Size Limit Rotation Policy" \
--set "retention-policy:File Count Retention Policy" \
--set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig set-connection-handler-prop \
--handler-name "HTTPS Connection Handler" \
```

```
--add "http-operation-log-publisher:HTTP Common Access"
```

- Optional. To support searching or filtering by DN using the Identity Access API, you can enable the entryDN virtual attribute. If running dsconfig interactive, open the main menu, select "Virtual Attribute" -> "View and edit an existing Virtual Attribute" -> "Entry DN", and then change the enabled property to "true". Click "finish" to save the setting.

```
$ bin/dsconfig set-virtual-attribute-prop --name entryDN --set enabled:true
```

- Optional. To support pagination, create some Virtual List View indexes. If running dsconfig interactive, open the main menu, select "Local DB VLV Index" -> "Create a new Local DB VLV Index" and then enter the properties needed for your setup. Click "finish" to save the setting. Repeat again for the "ascending-sn" index. Then, run the rebuild-index command to let the VLV Indexes take effect.

```
$ bin/dsconfig create-local-db-ylv-index --backend-name userRoot \
--index-name ascending-uid --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+uid"

$ bin/dsconfig create-local-db-ylv-index --backend-name userRoot \
--index-name ascending-sn --set base-dn:dc=example,dc=com \
--set scope:whole-subtree --set "filter:(objectclass=inetorgperson)" \
--set "sort-order:+sn"

$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.ascending-uid \
--index vlv.ascending-sn
```

To Enable Resource Versioning

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

- Enable the ds-entry-checksum virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
--name ds-entry-checksum \
--set enabled:true
```

- Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
--remove 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id ||
createTimestamp || ds-create-time || modifyTimestamp || ds-update-time")(version
3.0;acl "Authenticated read access to operational attributes used by the SCIM
servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

- Add new access controls required by SCIM for read access to operational attributes with the addition of the ds-entry-checksum:

```
$ bin/dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id ||
createTimestamp || ds-create-time || modifyTimestamp || ds-update-time || ds-entry-
checksum")(version 3.0;acl "Authenticated read access to operational attributes used
by the SCIM servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

- Enable SCIM resource versioning using the entry checksum virtual attribute:

```
$ bin/dsconfig set-http-servlet-extension-prop \
```



```
--extension-name SCIM \  
--set entity-tag-ldap-attribute:ds-entry-checksum
```

If enabled, the value of the `ds-entry-checksum` attribute is returned as the `ETag` header value when accessing the resource through SCIM, and is checked against the `If-Match` header when updating the resource. When accessing the resource through LDAP, use the `ds-entry-checksum` attribute instead.

To Configure the SCIM Servlet Extension using the Batch Script

The following example procedure assumes that you have set up your certificates, keystore, and truststore

1. Open the `<server-root>/config/scim-config-ds.dsconfig` script in a text editor.
2. For the optional elements (W3C common log, filtering by DN, and VLV Indexes, remove the comment (#) symbol on the `dsconfig` commands. Save the file when finished editing.
3. To enable the SCIM servlet extension, run the `dsconfig` batch file. Remember to include the bind parameters.

```
$ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
```

SCIM Servlet Extension Authentication

The SCIM servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM servlet attempts to correlate the username component of the Authorization header to a DN in the Directory Server. If the username value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingData provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

To Configure Basic Authentication Using an Identity Mapper

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the `uid` attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using `cn` values.

1. Create a new Identity Mapper that uses a match attribute of `cn`.

```
$ bin/dsconfig create-identity-mapper \  
--mapper-name "CN Identity Mapper" \  
--match-attribute cn
```

```
--type exact-match \  
--set enabled:true \  
--set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \  
--extension-name SCIM \  
--set "identity-mapper:CN Identity Mapper"
```

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \  
--handler-name ExampleOAuthTokenHandler \  
--type third-party \  
--set extension-class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \  
--extension-name SCIM \  
--set oauth-token-handler:ExampleOAuthTokenHandler
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

To Verify the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM resource URL via the command line or through a browser window.

- Run `curl` to verify that the SCIM extension is running. The `-k` (or `--insecure`) option is used to turn off `curl`'s verification of the server certificate, since the example Directory Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \  
-k "https://localhost:8443/scim/ServiceProviderConfigs"  
  
{ "schemas": ["urn:scim:schemas:core:1.0"], "id": "urn:scim:schemas:core:1.0",  
  "patch": {"supported": true}, "bulk": {"supported": true, "maxOperations": 10000,  
  "maxPayloadSize": 10485760}, "filter": {"supported": true, "maxResults": 100},  
  "changePassword": {"supported": true}, "sort": {"supported": true},  
  "etag": {"supported": false}, "authenticationSchemes": [{"name": "HttpBasic",  
  "description": "The HTTP Basic Access Authentication scheme. This scheme is  
  not considered to be a secure method of user authentication (unless used in  
  conjunction with some external secure system such as SSL), as the user  
  name and password are passed over the network as cleartext.", "specUrl":  
  "http://www.ietf.org/rfc/rfc2617", "documentationUrl":  
  "http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring Advanced SCIM Extension Features

The following sections show how to configure advanced SCIM servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM Schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DN's of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using `"dataType=binary"` in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the `<resource>` Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.

- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the `<attribute>` Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is `false`.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
 - `simple`: specifies a simple, singular SCIM attribute.
 - `complex`: specifies a complex, singular SCIM attribute.
 - `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
 - `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the `<simple>` Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `decimal`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the `<complex>` Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the `<simpleMultiValued>` Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is `email`.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is `false`.

A `simpleMultiValued` element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the `<complexMultiValued>` Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is `address`.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the `<subAttribute>` Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the `complexMultiValued` element.

About the `<canonicalValue>` Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, `work` is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the `<mapping>` Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the [Mapping LDAP Entries to SCIM Using the SCIM-LDAP API](#) section.

About the `<subMapping>` Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are

described later. The available transformations are described in [Mapping LDAP Entries to SCIM Using the SCIM-LDAP API](#).

About the <LDAPSearch> Element

An LDAPSearch element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DN's to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The LDAPSearch element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.



Note: The LDAPSearch element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The resourceIDMapping element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or `Directory Server`, meaning that a value is automatically provided by the Directory Server (as would be the case if the mapped LDAP attribute is `entryUUID`).

The following example illustrates an LDAPSearch element that contains a resourceIDMapping element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.

- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the `<fixedAttribute>` Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value `merge` indicates that the fixed values should be merged with the existing values. The value `overwrite` indicates that the existing values are to be overwritten by the fixed values. The value `preserve` indicates that no changes should be made. The default value is `merge`.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The PingData SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the Ping Identity Directory Server Administration Guide.



Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the <resourceIDMapping> Element".

Using Pre-defined Transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM `dateTime` data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the \$ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP Entries to SCIM Using the SCIM-LDAP API

In addition to the SCIM SDK, PingData provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
  <groupId>com.unboundid.product.scim</groupId>
  <artifactId>scim-ldap</artifactId>
  <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.



Note: The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP `DirectoryString` is automatically mapped to a SCIM string.

SCIM Authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM Logging

The Directory Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "origin=scim" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191  
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODFIY REQUEST conn=-3 op=190 msgID=191  
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM Monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "user-patch-404:26".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the `dsconfig` tool.

- **include-ldap-objectclass.** Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Server LDAP Schema.
- **exclude-ldap-objectclass.** Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn.** Specifies a multi-valued property that lists the base DN's that will be exposed. If specified, only entries under these base DN's will be accessible. No parent-child relationships in the DN's are allowed here.
- **exclude-ldap-base-dn.** Specifies a multi-valued property that lists the base DN's that will not be exposed. If specified, entries under these base DN's will not be accessible. No parent-child relationships in the DN's are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

To Configure the Identity Access API

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \  
  --extension-name SCIM --set 'include-ldap-objectclass:*' \  
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

To Disable Core SCIM Resources

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.



Note: When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using `dsconfig`, the changes take effect immediately without any restart required.

To Verify the Identity Access API Configuration

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
-H "Accept: application/json" \
"https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
attributes=entryDN,uid,givenName,sn,entryUUID"
{"schemas":["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
"id":"56c9fd6b-f870-35ef-9959-691c783b7318",
"meta":{"lastModified":"2013-01-11T23:38:26.489Z",
"location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
"urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":["user.1"],
"sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
"entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM Servlet Extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the `--resourceID` argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll \
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error `"java.net.SocketException: Too many open files"` if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

- > Number of successful operations per request type (such as GET, PUT, and POST).
- > Number of failed operations and their error codes per request type.
- > Number of operations with XML or JSON from client.
- > Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an `ldapsearch` such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
--bindPassword password --baseDN cn=monitor \
--searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called `host`.

```
$ bin/ldapsearch --baseDN cn=monitor \
'(objectclass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
```

```
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
```

```
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

About the HTTP Log Publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'[dd/MM/yyyy:HH:mm:ss Z]'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Chapter

24 Managing Server SDK Extensions

The Ping Identity Directory Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

Topics:

- [About the Server SDK](#)
- [Available Types of Extensions](#)

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.



Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available Types of Extensions

The Server SDK provides support for creating a number of different types of extensions for PingData Server Products, including the Ping Identity Directory Server, Ping Identity Directory Proxy Server, and Data Sync Server. Some of those extensions include:

Cross-Product Extensions

- > Access Loggers
- > Alert Handlers
- > Error Loggers
- > Key Manager Providers
- > Monitor Providers
- > Trust Manager Providers
- > OAuth Token Handlers
- > Manage Extension Plugins

Ping Identity Directory Server Extensions

- > Certificate Mappers
- > Change Subscription Handlers
- > Extended Operation Handlers
- > Identity Mappers
- > Password Generators
- > Password Storage Schemes
- > Password Validators
- > Plugins
- > Tasks

- Virtual Attribute Providers

Ping Identity Directory Proxy Server Extensions

- LDAP Health Checks
- Placement Algorithms
- Proxy Transformations

Data Sync Server Extensions

- JDBC Sync Sources
- JDBC Sync Destinations
- LDAP Sync Source Plugins
- LDAP Sync Destination Plugins
- Sync SourcesSync Destinations
- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Chapter

25

Troubleshooting the Server

The Ping Identity Directory Server provides a highly-reliable service that satisfies your company's objectives. However, if problems do arise (whether from issues in the Directory Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

This chapter provides information about how to perform this analysis to help ensure that the problem is resolved as quickly as possible. It targets cases in which the Directory Server is running on Sun Solaris or Linux systems, but much of the information can be useful on other platforms as well. This chapter presents the following information:

Topics:

- [*Working with the Collect Support Data Tool*](#)
- [*Directory Server Troubleshooting Information*](#)
- [*About the Monitor Entries*](#)
- [*Directory Server Troubleshooting Tools*](#)
- [*Troubleshooting Resources for Java Applications*](#)
- [*Troubleshooting Resources in the Operating System*](#)
- [*Common Problems and Potential Solutions*](#)

Working with the Collect Support Data Tool

The Directory Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for Solaris and Linux machines, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. For example, on Solaris Zone, configuration information is gathered using commands like `zonename` and `zoneadm`. However, the tool always tries to get the same information across all systems for the target Directory Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Server Commands Used in the Collect Support Data Tool

The following presents a summary of the data collectors that the `collect-support-data` tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

Table 65: Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
<code>status</code>	Runs <code>status -F</code> to show the full version information of the Directory Server (Unix, Windows).
<code>server-state</code>	Runs <code>server-state</code> to show the current state of the Directory Server process (Unix, Windows).
<code>dsreplication status</code>	Runs <code>dsreplication status</code> to show the status of the replicated topology (Unix, Windows). If the <code>--noReplicationStatus</code> option is used, the replication status information is not collected.

JDK Commands Used in the Collect-Support-Data Tool

Table 66: JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
<code>jps</code>	Java Virtual Machine Process status tool. Reports information on the JVM (Solaris, Linux, Windows, Mac OS).
<code>jstack</code>	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Solaris, Linux, Windows, Mac OS).
<code>jstat</code>	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Solaris, Linux, Windows, Mac OS).

Data Collector	Description
jinfo	Displays the Java configuration information for the Java process (Solaris, Linux, Windows, Mac OS).

Linux Commands Used in the collect-support-data Tool

Table 67: Linux Commands Used in the Collect-Support-Data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the <code>/var/logs/messages</code> directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: <ul style="list-style-type: none"> > <code>/proc/cpuinfo</code> > <code>/proc/meminfo</code> > <code>/etc/hosts</code> > <code>/etc/nsswitch.conf</code> > <code>/etc/resolv.conf</code>
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active process specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

Solaris Commands Used in the collect-support-data Tool

Table 68: Solaris Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
zonename	Prints the name of the current zone.
zoneadm	Prints the name of the current configured in verbose mode.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
zfs	Prints basic ZFS information: dataset pool names, and their used, available, referenced, and mountpoint properties.
zpool	Print a zpool's status.
fmdump	Prints the log files managed by the Solaris Fault Manager.
prtconf	Prints the system configuration information.
iostat	Prints disk I/O and CPU utilization information.
prtdiag	Prints the system diagnostic information.
cat	Concatenates the following files and prints to standard output: <ul style="list-style-type: none"> > <code>/proc/cpuinfo</code> > <code>/proc/meminfo</code>

Data Collector	Description
	<ul style="list-style-type: none"> > /etc/hosts > /etc/nsswitch.conf > /etc/resolv.conf
tail	Displays the last few lines of a file. Tails the /var/logs/messages directory and the /var/log/system.log directory.
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
patchadd	Prints the patches added to the system if any (Solaris, not OpenSolaris).
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active process specified by the pid.
prstat	Prints resource usage.

AIX Commands Used in the collect-support-data Tool

Table 69: AIX Commands Used in the Collect-Support-Data Tool

Data Collector	Description
ulimit	Defines user and system resources.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
zonename	Prints the name of the current zone.
cat	Concatenates the following files and prints to standard output: <ul style="list-style-type: none"> > /proc/cpuinfo > /proc/meminfo > /etc/hosts > /etc/nsswitch.conf > /etc/resolv.conf
vmstat	Prints information about virtual memory statistics.
alog	Prints the contents of the boot log file.
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sar	Print the local activity of the server.
lparstat	Prints logical partition information and statistics.
vmo	Prints the characteristics of one or more tunable parameters.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.

MacOS Commands Used in the Collect Support Data Tool

Table 70: MacOS Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.

Data Collector	Description
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the <code>/var/log/system.log</code> directory.
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sample	Profiles a process during an interval.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectory --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Server Troubleshooting Information

The Directory Server has a comprehensive default set of log files and monitor entries that are useful when troubleshooting a particular server problem.

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.

- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the `NOTICE`, `SEVERE_WARNING`, `SEVERE_ERROR`, and `FATAL_ERROR` severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the `PROTOCOL` category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the `DEBUG` severity for a category will not automatically enable logging at the `INFORMATION`, `MILD_WARNING`, or `MILD_ERROR` severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and

shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.



Note: Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Replication Repair Log

The replication repair log is written to `logs/replication` by default and records information about processing performed by the replication repair service. This log is used to resolve replication conflicts that can arise. For example, if the same entry is modified at the same time on two different systems, or if an attempt is made to create entries with the same DN at the same time on two different systems, the Directory Server records these events.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1" to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config" clientConnectionPolicy="default"
```

```
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensitive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN** – DN of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN** – DN of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The Commercial Edition of the LDAP SDK provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify

DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The Ping Identity Directory Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **`logs/tools/import-ldif.log`**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

je.info and je.config Files

The primary data store used by the Directory Server is the Oracle Berkeley DB Java Edition (JE). The Directory Server provides two primary sources of information about processing within the database.

The first is logging performed by the JE code itself, and is written into the `je.info.0` file in the server containing the database files (for example, `db/userRoot/je.info.0`). In the event of a problem within JE itself, useful information about the nature of the problem may be written to this log. The level of information written to this log file is controlled by the `db-logging-level` property in the backend configuration object. It uses the standard Java logging framework for logging messages, so the standard SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST levels are available.

The second is configuration information used when opening the database environment. When the backend database environment is opened, then the Directory Server will also write a file named `je.config` in the server containing the database files (for example, `db/userRoot/je.config`) with information about the configuration used.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

About the Monitor Entries

While the Directory Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database environment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides information about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.

- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Directory Server Troubleshooting Tools

The Ping Identity Directory Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Server software that is in use. If the server is running, then this information can be obtained from the "cn=Version,cn=monitor" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF Connection Handler

The Directory Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

dbtest Tool

The `dbtest` tool provides a utility that can be used to obtain general information about the data in a backend database. The tool dumps information about entries or keys, and raw data from the database. It can also find keys that have exceeded the entry limit.

For example, the following command can be used to dump a list of all keys in the `objectClass.equality` that have exceeded the entry threshold:

```
$ bin/dbtest dump-database-container \
  --backendID userRoot \
  --baseDN "dc=example,dc=com" \
  --databaseName objectClass.equality \
  --onlyExceedingLimit
```

On a large database, many `dbtest` operations may take a long time to complete, since every record in the associated database is examined. Use the database name option to list a specific database. The following command displays information about the `uid.equality` database in the `dc=example,dc=com` entry container in the `userRoot` backend.

```
$ bin/dbtest list-database-containers -n userRoot -b "dc=example,dc=com" -d uid.equality
```

Index Key Entry Limit

Indexes have keys that maintain a list of matching entries, up to the index entry limit. When that limit is reached, the key will not contain or maintain that list, and will just maintain a count of matching entries. To determine if index keys are approaching their limit, use either the `dbtest` tool or the `verify-index` tool.

While the `dbtest` tool can be used to gather general information, the `verify-index` tool provides statistical data about the percent of entries covered by the keys.

For example, the following command can be used to retrieve a list of keys that have exceeded the entry threshold:

```
$ bin/verify-index \
  --baseDN dc=example,dc=com \
  --listKeysExceedingIndexEntryLimit
```

The following is a sample of the data returned:

```
[12:06:05] Checked 6003 entries and found 0 error(s) in 2 seconds (average rate
2453.2/sec)
```

```

[12:06:05] Statistics for records that have exceeded the entry limit:
[12:06:05] The st.equality index has 48 such record(s) limit=100 min=103 max=152
median=118
[12:06:05] 1. or (152 entries / 2.53% of all entries)
[12:06:05] 2. ma (132 entries / 2.20% of all entries)
.....
[12:06:05] The id2subtree index has 2 such record(s) limit=4000 min=6000 max=6002
median=6001
[12:06:05] 1. 1 => dc=example,dc=com (6002 entries / 99.98% of all entries)
.....
[12:06:05] The id2children index has 1 such record(s) limit=4000 min=6000 max=6000
median=6000
[12:06:05] 1. 2 => ou=People,dc=example,dc=com (6000 entries / 99.95% of all entries)
[12:06:05] The objectClass.equality index has 4 such record(s) limit=4000 min=6001
max=6003 median=6001
[12:06:05] 1. top (6003 entries / 100.00% of all entries)
.....
    
```

Embedded Profiler

If the Directory Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the `--fileName` option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the `--fileName` option. To invoke GUI mode, add the option `--useGUI`.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Oracle Berkeley DB Java Edition Utilities

The Oracle Berkeley DB Java Edition (JE) itself provides a number of utilities that can be used for performing various types of low-level debugging in the database environment. These utilities should generally not be used unless you are advised to do so by your authorized support provider, but they provide access to information about the underlying database environment that is not available through any other means.

Troubleshooting Resources for Java Applications

Because the Ping Identity Directory Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the Ping Identity Directory Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- `-v` – Includes the arguments passed to the JVM for the processes that are listed.

- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The `jstack` tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the Ping Identity Directory Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where `{processID}` is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the `java` command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The `-l` option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The `-m` option can be used to include native frames in the stack trace.

jmap

The `jmap` tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native `pmap` tool provided by many operating systems. As with the `jstack` tool, `jmap` can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the `jhat` tool

- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the “:live” portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Server encounters an out of memory error (as a result of the “-XX:+HeapDumpOnOutOfMemoryError” JVM option) or from the use of the `jmap` command with the “-dump” option.

The `jhat` tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The `jhat` tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the `jhat` web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the “-port” option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The `jstat` tool is used to obtain a variety of statistical information from the JVM, much like the `vmstat` utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The `{interval}` option specifies the length of time in milliseconds between lines of output. The `{processID}` option specifies the process ID of the JVM used to run the Directory Server, which can be obtained by running `jps` as mentioned previously. The `{type}` option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.

- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Java Troubleshooting Tools (IBM JDK)

The Ping Identity Directory Server can be run on machines using the IBM JDK. IBM provides Java monitoring and diagnostic tools that can assess JVM performance and troubleshoot any Java application failures. The following tools are available for the IBM JDK. For more detailed information, see the IBM Developers web-site for a description of each tool:

- **Health Center Version 1.3.** Monitors Java applications running on the JDK. The tool provides profiling information for performance, memory usage, system environment, object allocations and other areas.
- **Memory Analyzer Version 1.1.** Analyzes Java heap memory using a system or heap dump snapshot of a Java process.
- **Garbage Collection and Memory Visualizer Version 2.6.** Fine-tunes Java performance by optimizing garbage collection performance, provides Java heap recommendations based on peak and average memory usage, and detects memory leaks and heap exhaustion.
- **Dump Analyzer Version 2.2.** Helps troubleshoot the cause of any application failure using an operating system dump. The tool detects any potential problems based on state, thread, stack information and error messages that were generated when the application failed.
- **Diagnostics Collector Version 1.0.** Collects diagnostic and context information during Java runtime processes that failed. The tool verifies your Java diagnostic configuration to ensure that disabled diagnostic analyzers are enabled to troubleshoot a problem.
- **IBM Diagnostic Tool Framework for Java Version 1.5.** Runs on dump data extracted by the `jextract` tool. The tool checks memory locations, Java threads, Java objects and other important diagnostic areas when the system dump was produced.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. Places to look for problems in the underlying system include:

- The system log file (`/var/adm/messages` on Solaris and `/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.
- On Solaris systems, if the fault management system has detected a problem with a system component, information about that problem is obtain by running the `fmddump` command.
- If the ZFS filesystem is in use, then the `zpool status` command provides information about read errors, write errors, or data checksum errors.

Examining CPU Utilization

Observing CPU utilization for the Directory Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere

(for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add `-P ALL` to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but `mpstat` reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use the `prstat` tool on Solaris or the `top` utility on Linux. If a process other than the Java process used to run the Directory Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Server to run effectively.

If the `mpstat` command showed that one CPU was much more heavily utilized than the others, it might be useful to identify the thread with the highest CPU utilization as it is likely the one that is a bottleneck preventing other threads from processing. On Solaris, this can be achieved by using the `prstat` command with the `-L` option, like:

```
prstat -L -p {processID}
```

This command will cause each thread to be displayed on a separate line, with the LWPID (lightweight process identifier) displayed as the last item on each line, separated from the process name by a slash. The thread that is currently consuming the largest amount of CPU will be displayed at the top of the list, and the `pstack` command can be used to identify which thread is responsible.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The `iostat` tool may be used to obtain information about the disk activity on the system. On Solaris systems, this should be invoked using the `-x` and `-n` arguments, like:

```
iostat -x -n 5
```

On Linux systems, `iostat` should be invoked with the `-x` argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the `%b` column on Solaris and the `%util` column on Linux. Both of these fields show the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a `tmpfs` filesystem).

On Solaris systems using ZFS, you can use the `zpool iostat {interval}` command to obtain information about I/O activity on a per-pool basis. While this command provides a useful display of the number of read and write operations and the amount of data being read from and written to the disks, it does not actually show how busy the underlying disks. As a result, the `zpool iostat` command is generally not as useful as the traditional `iostat` command for identifying potential I/O bottlenecks.

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard `ps` tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard `ps` command displays only a limited set of the arguments based on available space in the terminal window. In that case, the BSD version of the `ps` command (available on Solaris as `/usr/ucb/ps`) can be used to obtain the full command with all arguments, like:

```
/usr/ucb/ps auxww {processID}
```

pstack

The `pstack` command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using `jstack`, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the `pstack` command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like `pstack`, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger `gdb` is widely-used on Linux systems and is available on Solaris, but the Sun Studio debugger `dbx` is generally preferred over `gdb` on Solaris.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets) on Solaris, you can use the `pfiles` command, like:

```
pfiles {processID}
```

On Linux systems, the `lsof` tool can be used, like:

```
lsof -p {processID}
```

Tracing Process Execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot like you can get with `pstack` or a debugger. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

On Solaris systems, the `dtrace` tool provides an unmatched mechanism for tracing the execution of a process in extremely powerful and flexible ways, but it is also relatively complex and describing its use is beyond the scope of this document. In many cases, however, observing the system calls made by a process can reveal a great deal about what it is doing. This can be accomplished using the `truss` utility on Solaris or the `strace` tool on Linux.

The `truss` utility is very powerful and has a lot of options, but two of the most useful forms in which it may be invoked are:

- **`truss -f -p {processID}`** – Provides a basic overview of all system calls being made by the specified process (and any subprocesses that it creates) and their associated return values.
- **`truss -fear all -p {processID}`** – Provides an extremely verbose trace of all system call activity, including details about data being read from or written to files and sockets.

In both cases, the output may be written to a file instead of the terminal window by adding the `-o {path}` option. Further, rather than observing an already-running process, it is possible to have `truss` launch the process and trace execution over its entire life span by replacing `-p {processID}` with name and arguments for the command to invoke.

On Linux systems, the basic equivalent of the first `truss` variant above is:

```
strace -f -p {processID}
```

Consult the `strace` manual page for additional information about using it to trace process execution on Linux.

Problems with SSL Communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider
  \
  --set debug-level:verbose \
  --set include-throwable-cause:true

$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-ds` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining Network Communication

Because the Ping Identity Directory Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the `snoop` tool on Solaris or the `tcpdump` tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following commands can be used on Solaris and Linux, respectively:

```
snoop -d {interface} -r -x 0 host 1.2.3.4 port 389
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

On Solaris systems, the `snoop` command provides enhanced support for parsing LDAP communication (but only when the Directory Server is listening on the default port of 389). By adding the `-v` argument to the `snoop` command line, a verbose breakdown of each packet will be displayed, including protocol-level information. It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, in either case it is possible to write capture data to a file rather than displaying information on the terminal (using `-o {path}` with `snoop`, or `-w {path}` with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access. On Solaris systems, granting the `net_rawaccess` privilege to a user should be sufficient to allow that user to run the `snoop` utility.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, PingData recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the Administrative Console. The `status` tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:
 - > `logs/errors`
 - > `logs/failed-ops`
 - > `logs/expensive-ops`
3. Use system commands, such as `vmstat` and `iostat` to determine if the server is bottlenecked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the `periodic-stats-logger` can help to isolate problems, because it stores important server performance information on a per-second basis. The `periodic-stats-logger` can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".

5. For replication problem, run `dsreplication status` and look at the `logs/replication` file.
6. For more advanced users, run the `collect-support-data` tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the PingData Platform do not have direct access to the systems where the production servers are running. They can examine the `collect-support-data` archive on a different server. For more information, see [Using the Collect Support Data Tool](#).



Important: Run the `collect-support-data` tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server Will Not Run Setup

If the `setup` tool does not run properly, some of the most common reasons include the following:

A Suitable Java Environment Is Not Available

The Ping Identity Directory Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Oracle Berkeley DB Java Edition Is Not Available

If the version of the Directory Server that you are using was not provided with the Oracle Berkeley DB Java Edition library, then it must be manually downloaded and the appropriate JAR file placed in the `lib` directory before running `setup`. See the `lib/downloading-je.txt` file for instructions on obtaining the appropriate library.

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the `java` command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running `setup`, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Server prior to running the `setup` tool. This protects an existing Directory Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Server could not acquire an exclusive lock on file
/ds/PingDirectory/locks/server.lock: The exclusive lock requested for file
```

```
/ds/PingDirectory/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Server, or reconfigure the Directory Server to reduce the amount of memory that it attempts to use.
- The Directory Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- For Solaris-based systems only, if the system has one or more ZFS filesystems (even if the Directory Server itself is not installed on a ZFS filesystem), but it has not been configured to limit the amount of memory that ZFS can use for caching, then it is possible that ZFS caching is holding onto a significant amount of memory and cannot release it quickly enough when it is needed by the Directory Server. In that case, the system should be re-configured to limit the amount of memory that ZFS is allowed to use as described in the Using the Collect Support Data Tool.

- If the system is configured with one or more memory-backed filesystems, for example, `tmpfs` used for `/tmp` for Solaris), then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.
- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an `UNBOUNDID_JAVA_HOME` or `UNBOUNDID_JAVA_BIN` environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an `UNBOUNDID_JAVA_ARGS` environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the `UNBOUNDID_JAVA_HOME`, `UNBOUNDID_JAVA_BIN`, and `UNBOUNDID_JAVA_ARGS` environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-ds` command, but in most cases, none are required. If one or more command-line arguments were

provided for the `bin/start-ds` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Server configuration using an officially-supported tool like `dsconfig` or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the `--useLastKnownGoodConfig` option:

```
$ bin/start-ds --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.



Note: In addition to manually-editing the config file, you can look at previous archived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectory
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM

crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for Automatic Server Shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
  --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Server is not running.
- The underlying system on which the Directory Server is installed is not running.
- The Directory Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the `ldapsearch` command for example, use the `--useAdministrativeSession` option. The requester must have the `use-admin-session` privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.



Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.

- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the Administrative Console to see the Directory Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

```

--- Work Queue ---
      : Recent : Average : Maximum
-----:-----:-----:-----
Queue Size : 10  : 1      : 10
% Busy     : 17   : 14     : 100
    
```

You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see [Disk-Bound Deployments](#).
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat` should report very high utilization for disks used to hold swap space, and commands like `swap -l` on Solaris or `cat /proc/meminfo` on Linux can report a large amount of swap memory in use. Another cause of swapping is if `swappiness` is not set to 0 on Linux. For more information, see [Disable File System Swapping \(Linux\)](#).
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Server version 3.1 or later, the Directory Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.

- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server Must Disconnect a Client Connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the `cn=Active Operations,cn=monitor` monitor entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
--bindDN "cn=directory manager" \
--bindPassword password
```

- The monitor entry will contain attribute values for `operation-in-progress`, which look like an access log message. Look for the value of `conn` in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for `(description=expensive)`, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3 msgID=4
  clientIP="10.8.4.21" authDN="cn=appl,ou=applications,dc=example,dc=com" base="dc
  =example,dc=com" scope=wholeSubtree filter="(description=expensive)" attrs="A
  LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1 msgID=2
  clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root DNs,cn=config" base="c
  n=monitor" scope=wholeSubtree filter="(cn=active operations)" attrs="ALL"
num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named `disconnect6.ldif`.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name: com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to `ds-task-disconnect-connection-id`. The value for `ds-task-id` value does not follow a specific convention. It must be unique among other task entries currently cached by the server.

- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif \
--defaultAdd --bindDN "cn=directory manager" \
--bindPassword password
```

The Server is experiencing problems with replication

If replication does not appear to be functioning properly, then first check the `dsreplication status` command, which shows all of the servers that are replicating and whether they are backlogged or not. Next, you can check the server error log, replication repair log, and replication monitor entries may provide information about the nature of the underlying problem. Potential reasons that replication may not be functioning as expected include the following:

- Replication has not yet been configured between systems or has been disabled.
- If a server has been offline for a period of time or has fallen far enough behind such that it is missing changes, which are no longer present in any of the replication databases, then that server must be re-initialized with an up-to-date copy of the data from another server.
- If the environment is comprised of a heterogeneous set of systems, then it is possible that some of the systems might not be able to keep up with the maximum throughput achieved by other servers in the topology. In such cases, the slower servers might not be fast enough to remain in sync with the other servers in the environment.
- If the environment contains systems in multiple data centers and the network links between the data centers are insufficient for the volume of changes that must be processed, then servers might not be able to remain in sync under a high volume of changes.
- A network or firewall configuration problem has arisen, which prevents or interferes with communication between servers.
- An internal problem within the server has caused replication to stop functioning properly. The Directory Server logs the event in the error log in this case. Run the `collect-support-data` tool, so that the details of the problems can be passed to your authorized support provider. Then, try restarting the Directory Server.

How to Regenerate the Server ads-certificate

At setup time, the server generates a private key and certificate for use when secure communication between servers is required. This certificate, `ads-certificate`, is stored in `config/ads-truststore` and should typically remain unchanged for the life of the server deployment. If the need arises for a new `ads-certificate` to be created, say because the `server-root` has been copied to a new host, then the private key and certificate will be recreated by the startup process if the `config/ads-truststore` and `config/ads-truststore.pin` files are first manually removed while the server is offline. Note that if replication is enabled, the server must have replication disabled before regeneration of the `ads-certificate`.

For example, the Directory Server allows easy copying of its installation, which can then be used to install another server instance. If a server (`ldap1.example.com:389`) is enabled with its own copy (`ldap2.example.com:389`), `dsreplication` will exit with the following error message:

```
Replication cannot be enabled between servers ldap1.example.com:389 and ldap2.example.com:389 because they are using the same instance key.
```

The solution is to stop the server, remove `config/adtruststore` and `config/adtruststore.pin` and re-start the server. Upon startup, a new `adtruststore`, containing the server's instance key, will be generated. Then, you can re-run `dsreplication enable` to set up replication between the two servers.

The Server behaves differently from Sun/Oracle

After migrating from a Sun/Oracle configuration to an Ping Identity Directory Server, follow the tuning procedures in [Sun/Oracle Compatibility](#) if the Directory Server behaves differently from the Sun/Oracle server.

Troubleshooting ACI Evaluation

The Directory Server provides the ability to collect debug information related to ACI evaluation for any operation by enabling the Debug ACI Logger. The Debug ACI Logger is highly configurable and can be scoped to trace very specific request operations in order to narrow on any ACI issue that may arise in the field. Parameters for non-request operations, such as `log-connects`, `log-disconnects`, `log-security-negotiation`, `log-results`, `log-assurance-completed`, `log-search-entries`, `log-search-references`, `log-intermediate-responses` are set to `false` by default and should remain so.

Here is an example to enable the Debug ACI Logger:

```
$ bin/dsconfig set-log-publisher-prop \  
  --publisher-name "Debug ACI Logger" \  
  --set enabled:true
```

Using this debug tracer is often more efficient by limiting the output using request and result criteria to match specific types of operations. An example result criteria for operations that fail due to insufficient access rights can be added to the logger as follows:

```
$ bin/dsconfig set-log-publisher-prop \  
  --publisher-name "Debug ACI Logger" \  
  --set "result-criteria:Insufficient Access Rights"
```

Once the logger is enabled, all matching operations begin writing ACI evaluation traces to the log file. The amount of information is quite large for each evaluation that is done. However, this information is useful if there is an ACI issue that is difficult to resolve. Most operations result in multiple "ACI DEBUG" traces in the log, since it usually requires multiple ACI rights to perform an operation, each of which requires a separate evaluation. In particular, you can expect a lot of debug tracing when dealing with ACIs for controls, extended operations, and proxied authorization.

The ACI DEBUG traces contain the following pieces of information:

- **Operation.** Specifies a dump of the operation object that you can use to correlate to the original request operation.
- **ACI Container.** Specifies the context of the ACI evaluation being performed.
 - **Client Entry.** Specifies an LDIF dump of the client request access.
 - **Resource Entry.** Specifies an LDIF dump of the target resource.
 - **isProxiedAuth.** Specifies if the client is attempting to proxy as another user.
 - **Original Auth.** Specifies the original client DN if authorization is currently via the proxy.
 - **Rights.** Specifies a list of the ACI rights being requested on the resource entry.
 - **Control.** Specifies the OIDs when evaluating ACIs for a control.
 - **ExtOp.** Specifies the OIDs when evaluating ACIs for an extended operation.
- **ACI Candidates.** Specifies a list of all the ACIs known to this operation, sorted by origin.
- **Applicable ACIs.** Specifies a list of ACIs relevant to the current evaluation. These ACIs are separated by type into "Denies" and "Allows".
- **Deny ACI Evaluations.** Specifies the results of evaluating each "deny" ACI. If any of these evaluate to TRUE, then the operation will be denied.
- **Allow ACI Evaluations.** Specifies the results of evaluating each "allow" ACI. At least one of these must evaluate to "TRUE" or the operation will be denied.

For users with the `bypass-acl` privilege, the Debug ACI Logger will not provide any ACI debug tracing since evaluations are not done for those operations. However, you will see the following trace if you have ACI debugging enabled (`debug-aci-enabled` is set to TRUE) for those operations:

```
Bypassing ACL Evaluation for Operation
```

To avoid unnecessary tracing of these operations, the "Debug ACI Logger" uses a "Client Connection Criteria" called "Clients subject to Access Control" that excludes requests from users with the `bypass-acl` privilege. It is recommended that you create and use your own criteria which specifically targets the clients that you are trying to debug in order to make analyzing the tracing output easier.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Restricted Clients" \
  --type simple \
  --set none-included-user-privilege:bypass-acl
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.

- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM Memory Issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
error. Please report this problem and include this log file.
OutOfMemoryError: PermGen space
()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
(DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
(ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
(ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
(ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
(Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Problems with the HTTP Connection Handler

When problems with the HTTP Connection Handler occur, first look at the HTTP connection handler log to diagnose the issue. The following section shows HTTP log examples when various errors occur.

- **Failed Request Due to a Non-Existent Resource.** The server receives a status code 404, which indicates the server could not match the URI.

```
[15/Mar/2012:17:39:39 -0500] RESULT requestID=0 from="10.2.1.113:52958"
method="GET" url="https://10.2.1.113:443/Aleph/Users/uid=user.1,ou=people,
dc=example,dc=com" requestHeader="Host: x2270-11.example.lab"
requestHeader="Accept: */*" requestHeader="User-Agent: curl/7.21.6
(i386-pc-solaris2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22
libssh2/1.2.7" authorizationType="Basic" statusCode=404 etime=81.484
responseContentLength=103 responseHeader="Access-Control-Allow-Credentials:true"
responseContentType="application/json"
```

- **Failed Request due to a Malformed Request Body.** The server receives a status code 400, which indicates that the request had a malformed syntax in its request body.

```
[15/Mar/2012:17:47:23-0500] RESULT requestID=10 from="10.2.1.113:55284"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Expect: 100-continue"
requestHeader="Accept: */*" requestHeader="Content-Type: application/json"
requestHeader="User-Agent: curl/ 7.21.6 (i386-pc-solaris2.10) libcurl/7.21.6
OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7" authorizationType="Basic"
requestContentType="application/json" requestContentLength=5564 statusCode=400
etime=15.272 responseContentLength=133 responseContentType="application/json"
```

- **Failed Request due to an unsupported HTTP method.** The server receives a status code 405, which indicates that the specified method (e.g., "PATCH") in the request line is not allowed for the resource identified in the URI.

```
[15/Mar/2012:17:48:59-0500] RESULT requestID=11 from="10.2.1.113:55763"
method="PATCH" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-Type:"
```

```
application/json" requestHeader="User-Agent: curl/7.21.6 (i386-pc-solaris2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.2.2 libssh2/1.2.7"
authorization-Type="Basic" requestContentType="application/json" statusCode=405
etime=6.807 responseContentLength=0 responseHeader="Allow: POST, GET, OPTIONS, HEAD"
```

- **Failed Request due to an Unsupported Media Type.** The server receives a status code 415, which indicates that the request entity is in a format that is not supported by the requested resource.

```
[15/Mar/2012:17:44:45-0500] RESULT requestID=4 from="10.2.1.113:54493"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-Type:
application/atom+xml" requestHeader="User-Agent: curl/7.21.6 (i386-pc-solaris2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.2.2 libssh2/1.2.7"
authorizationType="Basic" requestContentType="application/atom+xml"
requestContentLength=3 statusCode=415 etime=6.222 responseContentLength=1402
responseHeader="Cache-Control: must-revalidate,no-cache,no-store"
responseContentType="text/html; charset=ISO-8859-1"
```

- **Failed Request due to an Authentication Error.** The server receives a status code 401, which indicates that the request requires user authentication.

```
[15/Mar/2012:17:46:06-0500] RESULT requestID=8 from="10.2.1.113:54899"
method="GET" url="https://10.2.1.113:443/Aleph/Schemas" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="User-Agent:
curl/7.21.6 (i386-pc-solaris2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5
libidn/1.2.2 libssh2/ 1.2.7" authorizationType="Basic" statusCode=401
etime=2.751 responseContentLength=63 responseHeader="WWW-Authenticate: Basic
realm=SCIM" responseHeader="Access-Control-Allow-Credentials: true"
responseContentType="application/json"
```

Virtual Process Size on RHEL6 Linux is Much Larger than the Heap

Red Hat Linux introduced a change in glibc 2.11 that creates larger per-thread address spaces aligned at 64MB. This change results in a virtual process size much larger than those seen in previous versions of glibc. This is not considered a bug by RedHat, as noted in https://bugzilla.redhat.com/show_bug.cgi?id=640286, and does not affect the physical memory needed by the server process. To see the version of glibc on your system, use the command `yum info glibc`.

AIX Command-Line Tools Not Working with DNs using Special Characters

The Ping Identity Directory Server command-line tools may not successfully run with base DNs that contain special characters (characters that require escaping) on AIX using some IBM Java versions.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run

this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter

26

Command-Line Tools

The Ping Identity Directory Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

This chapter presents the following topics:

Topics:

- [*Using the Help Option*](#)
- [*Available Command-Line Utilities*](#)
- [*Managing the `tools.properties` File*](#)
- [*Running Task-based Utilities*](#)

Using the Help Option

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. You can view detailed argument options and examples by typing `--help` with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (for example, `dsconfig`), you can get a list of the subcommands by typing `--help-subcommands`.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing `--help` with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```



Note: For detailed information and examples of the command-line tools, see the *PingData Directory Server Command-Line Tool Reference*.

Available Command-Line Utilities

The Directory Server provides the following command-line utilities, which can be run directly in interactive, non-interactive, or script mode.

Table 71: Command-Line Utilities

Command-Line Tools	Description
audit-data-security	Performs an internal task that examines all or a subset of entries in the server, writing a series of reports on potential risks with the data. Reports are written to the output directory organized by backend name and audit items.
authrate	Perform repeated authentications against the Directory Server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more directory server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
config-diff	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.

Command-Line Tools	Description
create-rc-script	Create a Run Control (RC) script that may be used to start, stop, and restart the Directory Server on UNIX-based systems.
dbtest	Inspect the contents of Directory Server backends that store their information in Oracle® Berkeley DB Java Edition databases.
deliver-one-time-password	Submit a "deliver one-time password" extended request, OID 1.3.6.1.4.1.30221.2.6.24, to the server which results in the generation of a one-time password which is delivered out-of-band to the specified user. This tool can be used to test the UNBOUNDID-DELIVERED-OTP SASL mechanism.
dsconfig	View and edit the Directory Server configuration.
dsframework	Manage administrative server groups or the global administrative user accounts that are used to configure servers within server groups.
dsjavaproperties	Configure the JVM arguments used to run the Directory Server and associated tools. Before launching the command, edit the properties file located in <code>config/java.properties</code> to specify the desired JVM options and <code>JAVA_HOME</code> environment variable.
dsreplication	Manage data replication between two or more Directory Server instances.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
encode-password	Encode user passwords with a specified storage scheme or determine whether a given clear-text value matches a provided encoded password.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Server enter lockdown mode, during which it only processes operations requested by users holding the <code>lockdown-mode</code> privilege.
export-ldif	Export data from the Directory Server backend in LDIF form.
identify-references-to-missing-entries	Identify entries containing one or more attributes that reference entries that do not exist. This may require the ability to perform unindexed searches and/or the ability to use the simple paged results control.
identify-unique-attribute-conflicts	Identify unique attribute conflicts. The tool may identify values of one or more attributes that are supposed to exist only in a single entry but are found in multiple entries.
import-ldif	Import LDIF data into the Directory Server backend.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Server.
ldapdelete	Perform LDAP delete operations in the Directory Server.
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Server.
ldapsearch	Perform LDAP search operations in the Directory Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Server leave lockdown mode and resume normal operation.

Command-Line Tools	Description
list-backends	List the backends and base DNs configured in the Directory Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-account	Access and alter password policy state properties for user entries.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the Ping Identity Directory Server. Extension bundles are installed from a zip archive or file system directory. Ping Identity Directory Server will be restarted if running to activate the extension(s).
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Server.
migrate-ldap-schema	Migrate schema information from an existing LDAP server into an Ping Identity Directory Server instance.
migrate-sun-ds-config	Update an instance of the Ping Identity Directory Server to match the configuration of an existing Sun Java System 5.x, 6.x, or 7.x directory instance.
modrate	Perform repeated modifications against a Directory Server.
move-subtree	Move a subtree entries or a single entry from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
profile-viewer	View information in data files captured by the Directory Server profiler.
re-encode-entries	Initiate a task that causes a local DB backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend (e.g., to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled).
rebuild-index	Rebuild index data within a backend based on the Berkeley DB Java Edition. Note that this tool uses different approaches to rebuilding indexes based on whether it is running in online mode (as a task) rather than in offline mode. Running in offline mode will often provide significantly better performance. Also note that running in online mode will prevent the server from using that index while the rebuild is in progress, so some searches may behave differently while a rebuild is active than when it is not.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Server backend.
restore	Restore a backup of the Directory Server backend.
revert-update	Returns a server to the version before the last update was performed.
review-license	Review and/or indicate your acceptance of the product license.
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Server process.
setup	Perform the initial setup for the Directory Server instance.
start-ds	Start the Directory Server.
status	Display basic server information.
stop-ds	Stop or restart the Directory Server.

Command-Line Tools	Description
subtree-accessibility	List or update the a set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
uninstall	Uninstall the Directory Server.
update	Update the Directory Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the <code>update</code> tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update is performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the <code>revert-update</code> tool for information on reverting an update.
validate-acis	Validates a set of access control definitions contained in an LDAP server (including Sun/Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the Directory Server. Note that the output generated by this tool will be in LDIF format, but each entry in the output will have exactly one ACI, so entries that have more than one ACI will appear multiple times in the output with different ACI values.
validate-file-signature	Validate the signature information in a signed text file, such as a signed log file or a signed LDIF export.
validate-ldif	Validate the contents of an LDIF file against the server schema.
verify-index	Verify that indexes in a backend using the Oracle Berkeley DB Java Edition are consistent with the entry data contained in the database.

Managing the tools.properties File

The Ping Identity Directory Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool's arguments. Properties files are convenient when quickly testing the Directory Server in multiple environments.

The Directory Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the `--propertiesFilePath` option. You can override all of the Directory Server's command-line utilities with a properties file using the `config/tools.properties` file.

Creating a Tools Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
```

```
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the `--propertiesFilePath /path/to/` File option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Another example using the `dsconfig` configuration tool is as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```



Note: The `.bindPasswordFile` property requires an absolute path. If you were to specify `~/ds/config/password`, where `~` refers to the home directory, the server does not expand the `~` value when read from the properties file.

Specifying Default Properties Files

The Directory Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, `tools.properties`, is located in the `<server-root>/config` directory.

If you place a custom properties file that has a different filename as `tools.properties` in this default location, you need to specify the path using the `--propertiesFilePath` option. If you

make changes to the `tools.properties` file, you do not need the `--propertiesFilePath` option. See the examples in the next section.

Evaluation Order Summary

The Directory Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the `--propertiesFilePath` option is used with no other options, the Directory Server takes its options from the specified properties file.
- If no options are used on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), the Directory Server searches for the `tools.properties` file at `<server-root>`
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) have precedence over general properties (for example, `port=1389`).

Evaluation Order Example

Given the following properties file that is saved as `<server-root>/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

The Directory Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the `bindDN` and `bindPassword` arguments specified in the properties file.

```
$ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
--propertiesFilePath bin/tools.properties "(objectclass=*)"
```

2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the Directory Server uses the specified properties file as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
"(objectclass=*)"
```

3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, the Directory Server attempts to locate any default `tools.properties` file in the following location:

```
<server-root>/config/tools.properties
```

Assume that you move your `tools.properties` file from `<server-root>/bin` to the `<server-root>/config` directory. You can then run your tools as follows:

```
$ bin/ldapsearch "(objectclass=*)"
```

The Directory Server can be configured so that it does not search for a `properties` file by using the `--noPropertiesFile` option. This option tells the Directory Server to use only those options specified on the command line. The `--propertiesFilePath` and `--noPropertiesFile` options are mutually exclusive and cannot be used together.

4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Running Task-based Utilities

The Directory Server has a Tasks subsystem that allows you to schedule basic operations, such as backup, restore, `bin/start-ds`, `bin/start-ds` and others. All task-based utilities require the `--task` option that explicitly indicates the utility is intended to run as a task rather than in offline mode. The following table shows the arguments that can be used for task-based operations:

Table 72: Task-based Utilities

Option	Description
<code>--task</code>	Indicates that the tool is invoked as a task. The <code>--task</code> argument is required. If a tool is invoked as a task without this <code>--task</code> argument, then a warning message will be displayed stating that it must be used. If the <code>--task</code> argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.
<code>--start</code>	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
<code>--dependency</code>	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
<code>--failedDependencyAction</code>	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: <code>PROCESS</code> , <code>CANCEL</code> , <code>DISABLE</code> . If not specified, the default value is <code>CANCEL</code> . This option can be used multiple times in a single command.
<code>--completionNotify</code>	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
<code>--errorNotify</code>	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.