opentext™

OpenText™ eDirectory CE 24.4 (v9.3)

Administration Guide

January 2025

Legal Notice

Copyright 2009 - 2024 Open Text All rights reserved.

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contents

About this Book and the Library 17

1 Understanding OpenText eDirectory 19

Ease of Management through OpenText Identity Console 20
- Powerful Tree Structure 20
- Web-Based Management Utility 22
- Single Login and Authentication 23

Object Classes and Properties 23
- List of Objects 23
- Container Object Classes 25
- Leaf Object Classes 29

Context and Naming 42
- Distinguished Name 43
- Typeful Name 43
- Name Resolution 43
- Current Workstation Context 43
- Leading Period 44
- Relative Naming 44
- Trailing Periods 44
- Context and Naming on Linux 45

Schema 45
- Schema Management 45
- Schema Classes, Attributes, and Syntaxes 46
- Understanding Mandatory and Optional Attributes 52
- Sample Schema 52
- Designing the Schema 53

Partitions 53
- Partitions 54
- Distributing Replicas for Performance 54
- Partitions and WAN Links 54

Replicas 56
- Replica Types 57
- Filtered Replicas 60

Server Synchronization in the Replica Ring 62

Access to Resources 62

OpenText eDirectory Rights 63
- Trustee Assignments and Targets 63
- OpenText eDirectory Rights Concepts 63
- Default Rights for a New Server 68
- Delegated Administration 69
- Administering Rights 69

2 Designing Your OpenText eDirectory Network 75

OpenText eDirectory Design Basics 75
- Network Layout 75
- Organizational Structure 76

Preparing for OpenText eDirectory Design 76
Designing the OpenText eDirectory Tree 76
Creating a Naming Standards Document 76
Designing the Upper Layers of the Tree 79
Designing the Lower Layers of the Tree 81
Guidelines for Partitioning Your Tree 82
Determining Partitions for the Upper Layers of the Tree 82
Determining Partitions for the Lower Layers of the Tree 83
Determining Partition Size 83
Considering Network Variables 83
Guidelines for Replicating Your Tree 84
Workgroup Needs 84
Fault Tolerance 84
Determining the Number of Replicas 85
Replicating the Tree Partition 85
Replicating for Administration 86
Managing WAN Traffic 86
Planning the User Environment 86
Reviewing Users' Needs 86
Creating Accessibility Guidelines 87
Designing OpenText eDirectory for e-Business 87
Understanding the OpenText Certificate Server 88
Rights Required to Perform Tasks on OpenText Certificate Server 88
Ensuring Secure OpenText eDirectory Operations on Linux Computers 89
Synchronizing Network Time 92
Synchronizing Time on Linux Computers 93
Verifying Time Synchronization 93

3 Managing Objects 95

General Object Tasks 95
Browsing the OpenText eDirectory Tree 96
Creating an Object 97
Modifying an Object's Properties 98
Copying Objects 98
Moving Objects 98
Deleting Objects 99
Renaming Objects 99
Managing User Accounts 99
Creating and Modifying User Accounts 100
Setting Up Optional Account Features 101
Disabling the Login Time Update Interval 103
Setting Up Login Scripts 104
Login Time Restrictions for Remote Users 105
Deleting User Accounts 106
Configuring Roles and Access Control 106
Defining RAC Roles 108
Defining Custom RAC Tasks 110

4 Managing Background Process 113

Synchronization 113
Features of Synchronization 114
Normal or Replica Synchronization 116

Priority Sync 118
Policy Based Replication 125
Manually Configuring Synchronization Threads 126
Configuring Asynchronous Outbound Synchronization 127
Configuring Background Processes 128
Hard Limit Policy 128
CPU-Based Dynamic Policy 128
Background Process Interval 128

5 Managing the Schema 131

Extending the Schema 132
Creating a Class 132
Deleting a Class 132
Creating an Attribute 133
Adding an Optional Attribute to a Class 133
Deleting an Attribute 134
Creating an Auxiliary Class 134
Extending an Object with the Properties of an Auxiliary Class 134
Modifying an Object's Auxiliary Properties 135
Deleting Auxiliary Properties from an Object 135
Viewing the Schema 135
Viewing Class Information 136
Viewing Attribute Information 136
Manually Extending the Schema 136
Extending the Schema on Windows 136
Extending the Schema on Linux 137
Schema Flags Added in version 8.7 Onwards 139
Using the Client to Perform Schema Operations 140
Using the DSSchema eMTool 140
DSSchema eMTool Options 141

6 Managing Partitions and Replicas 143

Creating a Partition 144
Merging a Partition 144
Moving Partitions 146
Canceling Create or Merge Partition Operations 147
Administering Replicas 147
Adding a Replica 147
Deleting a Replica 148
Changing a Replica Type 149
Setting Up and Managing Filtered Replicas 150
Using the Filtered Replica Wizard 151
Defining a Partition Scope 151
Setting Up a Server Filter 152
Viewing Partitions and Replicas 153
Viewing the Partitions on a Server 153
Viewing a Partition’s Replicas 153
Viewing Information about a Partition 154
Viewing Partition Hierarchy 154
Viewing Information about a Replica 154

7 OpenText eDirectory Management Utilities 157

OpenText Import Conversion Export Utility 157
Using the Command Line Interface 158
Conversion Rules 176
LDAP Bulk Update/Replication Protocol 185
Improving the Speed of LDIF Imports 185
Index Manager 187
Creating an Index 188
Deleting an Index 188
Taking an Index Offline 189
Managing Indexes on Other Servers 189
OpenText eDirectory Service Manager 189
Using the Client Service Manager eMTool 190
Offline Bulkload Utility 191
Improving Bulkload Performance 191
Using ldif2dib for Bulkloading 195
Multiple Instances 196
Tuning ldif2dib 196
Limitations 198
Caveats 199
LDIF Files 200
Understanding LDIF 200
Debugging LDIF Files 209
Using LDIF to Extend the Schema 211
ldif2dib Limitations 216

8 Monitoring OpenText 219

Using OpenText iMonitor 219
System Requirements 220
Accessing OpenText iMonitor 221
OpenText iMonitor Architecture 222
OpenText iMonitor Features 227
Ensuring Secure OpenText iMonitor Operations 249
Configuring HTTP Server Object 250
Setting HTTP Stack Parameters Using ndsconfig 251
Using cn=monitor for Monitoring 252
Viewing the Monitoring Statistics 253
Using DSTrace 263
Basic Functions 264
Debugging Messages 265
Background Processes 267
DSTrace Messages 271
Linux 272
Windows 273
OpenText iMonitor Message Filtering 275
SAL Message Filtering 275
Configuring the Severity Levels 275
Setting the Log File Path 276

9 SecretStore Configuration for OpenText eDirectory Server 277

Linux 277

Windows 278

10 Merging OpenText eDirectory Trees 279

Merging OpenText eDirectory Trees 279
Prerequisites 280
Target Tree Requirements 280
Schema Requirements 280
Merging the Source into the Target Tree 281
Partition Changes 281
Preparing the Source and Target Trees 282
Synchronizing Time before the Merge 282
Merging Two Trees 283
Post-Merge Tasks 284
Grafting a Single Server Tree 285
Understanding Context Name Changes 287
Preparing the Source and Target Trees 288
Containment Requirements for Grafting 289
Grafting the Source and Target Tree 290
Renaming a Tree 290
Using the Client to Merge Trees 291
Using the DSMerge eMTool 291
DSMerge eMTool Options 292

11 Encrypting Data in OpenText eDirectory 293

Encrypted Attributes 293
Using Encryption Schemes 295
Managing Encrypted Attributes Policies 295
Accessing the Encrypted Attributes 300
Viewing the Encrypted Attributes 301
Encrypting and Decrypting Backup Data 302
Cloning the DIB Fileset Containing Encrypted Attributes 302
Adding OpenText eDirectory Servers to Replica Rings 302
Backward Compatibility 302
Migrating to Encrypted Attributes 303
Replicating the Encrypted Attributes 303
Encrypted Replication 303
Need for Encrypted Replication 304
Enabling Encrypted Replication 304
Adding a New Replica to a Replica Ring 308
Synchronization and Encrypted Replication 309
Viewing the Encrypted Replication Status 309
Achieving Complete Security While Encrypting Data 310
Encrypting Data in an All New Setup 310
Encrypting Data in an Existing Setup 311
Conclusion 312

12 Repairing the OpenText eDirectory Database 313

Performing Basic Repair Operations 314
Performing an Unattended Full Repair 315
Performing a Local Database Repair 316
Checking External References 316

Repairing a Single Object 316
Deleting Unknown Leaf Objects 317
Viewing and Configuring the Repair Log File 317
Opening the Log File 317
Setting Log File Options 318
Performing a Repair in OpenText iMonitor 318
Repairing Replicas 318
Repairing All Replicas 318
Repairing Selected Replicas 319
Repairing Time Stamps 319
Designating This Server As the New Master Replica 320
Destroying the Selected Replica 320
Repairing Replica Rings 320
Repairing All Replica Rings 321
Repairing the Selected Replica Ring 321
Sending All Objects to Every Server in the Ring 321
Receiving All Objects from the Master to the Selected Replica 322
Removing This Server from the Replica Ring 322
Maintaining the Schema 322
Requesting Schema from the Tree 323
Resetting the Local Schema 323
Performing Optional Schema Enhancements 323
Importing Remote Schema 324
Declaring a New Schema Epoch 324
Repairing Server Network Addresses 325
Repairing All Network Addresses 325
Repairing a Server's Network Addresses 325
Performing Synchronization Operations 326
Synchronizing the Selected Replica on This Server 326
Reporting the Synchronization Status on This Server 327
Reporting the Synchronization Status on All Servers 327
Performing a Time Synchronization 327
Scheduling an Immediate Synchronization 328
DSRepair Options 328
Running DSRepair on the OpenText eDirectory Server 328
DSRepair Command Line Options 330
Using Advanced DSRepair Switches 332
Using the Client to Repair a Database 333
Using the DSRepair eMTool 333
DSRepair eMTool Options 334
Graphical DS Repair Utility 335

13 Understanding LDAP Services for OpenText eDirectory 337

Key Terms for LDAP Services 338
Clients and Servers 338
Objects 338
Referrals 339
Understanding How LDAP Works with OpenText eDirectory 341
Connecting to OpenText eDirectory from LDAP 341
Class and Attribute Mappings 344
Syntax Differences 347
Supported OpenText LDAP Controls and Extensions 349
Using LDAP Tools on Linux 349

LDAP Tools . . . 350
Extensible Match Search Filter . . . 361
LDAP Transactions . . . 362
Limitations . . . 364

14 Configuring LDAP Services for OpenText eDirectory 365

Loading and Unloading LDAP Services for OpenText eDirectory . . . 365
Verifying That the LDAP Server Is Loaded . . . 366
Verifying That the LDAP Server Is Running . . . 367
Scenarios . . . 367
Verifying That The LDAP Server Is Running . . . 368
Verifying That A Device Is Listening . . . 368
Preventing POODLE Attack by Disabling SSLv3 . . . 368
Configuring LDAP Objects . . . 369
Configuring LDAP Server and LDAP Group Objects on Linux . . . 370
Configuring Protocols and Ciphers Using `ldapSSLConfig` Attribute . . . 380
Refreshing the LDAP Server . . . 383
Authentication and Security . . . 384
Requiring TLS for Simple Binds with Passwords . . . 385
Starting and Stopping TLS . . . 385
Configuring the Server for TLS . . . 386
Configuring the Client for TLS . . . 387
Exporting the Trusted Root . . . 388
Authenticating with a Client Certificate . . . 389
Using Certificate Authorities from Third-Party Providers . . . 389
Creating and Using LDAP Proxy Users . . . 389
Using SASL . . . 391
Using NMAS Based Logins for LDAP Authentication . . . 393
Using the LDAP Server to Search the Directory . . . 393
Setting Search Limits . . . 393
Using Referrals . . . 394
Configuring for Superior Referrals . . . 403
Scenario: Superior Referrals in a Federated Tree . . . 403
Creating a Nonauthoritative Area . . . 404
Specifying Reference Data . . . 405
Updating Reference Information through LDAP . . . 406
Affected Operations . . . 406
Discovering Support for Superior References . . . 407
Persistent Search: Configuring for OpenText eDirectory Events . . . 407
Managing Persistent Searches . . . 408
Controlling Use of the Monitor Events Extended Operation . . . 408
Getting Information about the LDAP Server . . . 409
Configuring Generalized Time Support . . . 411
Configuring Permissive Modify . . . 411
Proxied Authorization Control . . . 412
LDAP Paged Search Control . . . 412
LDAP Extended DN Control . . . 413
Auditing LDAP Events . . . 416

15 Backing Up and Restoring OpenText eDirectory 417

Checklist for Backing Up OpenText eDirectory . . . 418

Understanding Backup and Restore Services 421
About the OpenText eDirectory Backup Tool 421
What's Different between Backup and Restore in DSBK and TSA for NDS Backup 422
Overview of How the Backup Tool Does a Restore 423
Format of the Backup File Header 424
Format of the Backup Log File 429
Using DSMASTER Servers as Part of Disaster Recovery Planning 430
Transitive Vectors and the Restore Verification Process 431
Using Roll-Forward Logs 432
Issues to Be Aware of When Turning On Roll-Forward Logging 433
Location of the Roll-Forward Logs 434
Backing Up and Removing Roll-Forward Logs 435
Cautionary Note: Removing OpenText eDirectory Also Removes the Roll-Forward Logs 436
Preparing for a Restore 436
Prerequisites for Restoring 436
Locating the Right Backup Files for a Restore 437
Using DSBK 439
Prerequisites 439
Using DSBK on Various Platforms 440
Backing Up Manually with DSBK 442
Automating the Backing Up of OpenText eDirectory 443
Configuring Roll-Forward Logs with DSBK 443
Restoring from Backup Files with DSBK 444
Backup and Restore Command Line Options 446
Running DSBK as a cron Job 454
Backing Up and Restoring NICI 454
Backing Up NICI 455
Restoring NICI 455
Recovering the Database If Restore Verification Fails 456
Cleaning Up the Replica Ring 457
Repair the Failed Server and Re-add Replicas to the Server 458
Scenarios for Backup and Restore 460
Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Single-Server NetWork 460
Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Multiserver Environment . . . 461
Scenario: Losing an Entire Server in a Multiple-Server Environment 463
Scenario: Losing Some Servers in a Multiple-Server Environment 464
Scenario: Losing All Servers in a Multiple-Server Environment 464
Disaster Recovery Plan using DSBK 466
Disaster Recovery Plan on Linux 466
Disaster Recovery Plan on Windows 467
LDAP-Based Backup 468
Need for LDAP Based Backup 469
For More Information 469
OpenText eDirectory Backup with SMS 469

16 Configuring OpenText eDirectory in Suite B Mode 471

Enabling Suite B in a New Installation 472
Enabling Suite B on the Certificate Server 473
Configuring LDAP and HTTP Services to Use ECDSA Certificates and Suite B Ciphers 474
Creating an AES 256-Bit SDI Key 476
Enabling Background Authentication 477
Configuring Suite B on Existing Servers 477

17 Enabling Enhanced Background Authentication 479

Enabling EBA . . . 481
Enabling EBA on an OpenText eDirectory Tree . . . 481
Enabling EBA on an OpenText eDirectory Server . . . 482
Disabling EBA on an OpenText eDirectory Server . . . 483
Viewing Information About EBA . . . 483
Managing the EBA CA by Using OpenText Identity Console . . . 485
Restrictions in OpenText eDirectory Operations When EBA Is Enabled . . . 485
Restrictions on Changing Replica Types . . . 485
Restrictions on Changing the Master of a Partition . . . 486
Restrictions on Merging Partitions . . . 486
Restrictions on Reconfiguring a Server Enabled with EBA . . . 486
Backing Up an EBA Enabled Server . . . 486
Moving the EBA CA Role to a New Server . . . 486

18 SNMP Support for OpenText eDirectory 489

Definitions and Terminology for SNMP . . . 489
Understanding SNMP Services . . . 490
OpenText eDirectory and SNMP . . . 492
Benefits of SNMP Instrumentation on OpenText eDirectory . . . 492
Understanding How SNMP Works with OpenText eDirectory . . . 492
Installing and Configuring SNMP Services for OpenText eDirectory . . . 494
Loading and Unloading the SNMP Server Module . . . 494
Subagent Configuration . . . 495
Setting Up SNMP Services for OpenText eDirectory . . . 497
Monitoring OpenText eDirectory Using SNMP . . . 500
Traps . . . 501
Configuring Traps . . . 515
Statistics . . . 522
Troubleshooting . . . 526

19 Maintaining OpenText eDirectory 529

Advanced Referral Costing . . . 529
Improving Server-to-Server Connection . . . 530
Advantages of Referral Costing . . . 532
Deploying ARC . . . 533
Enabling Advanced Referral Costing . . . 534
Tuning Advanced Referral Costing . . . 534
Monitoring Advanced Referral Costing . . . 535
Keeping OpenText eDirectory Healthy . . . 538
When to Perform Health Checks . . . 538
Health Check Overview . . . 539
Checking OpenText eDirectory Health Using OpenText iMonitor . . . 539
For More Information . . . 540
Resources for Monitoring . . . 541
Upgrading Hardware or Replacing a Server . . . 541
Planned Hardware or Storage Device Upgrade without Replacing the Server . . . 541
Planned Replacement of a Server . . . 544
Server IP Address Changes . . . 547
Restoring OpenText eDirectory after a Hardware Failure . . . 547

Subtree Search Performance Improvement . . . 548
Container Readiness . . . 548

20 DHost OpenText iConsole Manager 549

What is DHost? . . . 550
Running DHost OpenText iConsole . . . 550
Running DHost OpenText iConsole on Windows . . . 551
Running DHost OpenText iConsole on Linux . . . 551
Managing OpenText eDirectory Modules . . . 551
Loading or Unloading Modules on Windows . . . 552
Loading or Unloading Modules on Linux . . . 552
Querying for DHost Information . . . 553
Viewing the Configuration Parameters . . . 553
Viewing Protocol Information . . . 553
Viewing Connection Properties . . . 554
Viewing the Thread Pools Statistics . . . 554
Process Stack . . . 555

21 Setting the sadmin Password 557

22 The OpenText eDirectory Management Toolbox 559

Using the Command Line Client . . . 560
Displaying the Command Line Help . . . 561
Running the Command Line Client in Interactive Mode . . . 561
Running the Command Line Client in Batch Mode . . . 565
eMBox Command Line Client Options . . . 567
Establishing a Secure Connection with the Client . . . 568
Finding Out OpenText eDirectory Port Numbers . . . 568
Using the Logger . . . 569
Using the Logger Command Line Client . . . 569
Using the eMBox Client for Backup and Restore . . . 570
Prerequisites . . . 571
Backing Up Manually with the eMBox Client . . . 572
Doing Unattended Backups, Using a Batch File with the eMBox Client . . . 573
Configuring Roll-Forward Logs with the eMBox Client . . . 574
Restoring from Backup Files with the eMBox Client . . . 576

23 Auditing OpenText eDirectory Events 579

Auditing with CEF . . . 579
Configuring CEF . . . 579
Journal Event Caching . . . 600
LDAP Auditing . . . 601
Need for LDAP Auditing . . . 601
Using LDAP Auditing . . . 601
For More Information . . . 602

24 Understanding OpenText eDirectory's Authentication Framework 603

NMAS Functionality . . . 603
User Identification Phase . . . 603

Authentication (Login) Phase . . . 604
Device Removal Detection Phase . . . 605
Login and Post-Login Methods and Sequences . . . 606
Security Object Caching . . . 606
NMAS Software . . . 607
Server and Client Software Installation . . . 607
Login Method Software and Partners . . . 608
Universal Password . . . 608
OpenText Identity Console Management . . . 609
Managing Login and Post-Login Methods and Sequences . . . 609
Ways of Installing a Login Method . . . 610
Updating Login and Post-Login Methods . . . 611
Managing Login Sequences . . . 611
Authorizing Login Sequences for Users . . . 613
Setting Default Login Sequences . . . 614
Deleting a Login Method . . . 614
Deleting a Login Sequence . . . 615
Using NMAS to Log In to the Network . . . 616
Password Field . . . 616
Advanced Login . . . 616
Unlocking the Workstation . . . 617
Capturing an NMAS Client Trace . . . 617
Viewing NMAS Clearance Status . . . 617
History of OpenText Passwords . . . 617
NMAS HOTP Based Login . . . 618
Overview . . . 618
Installation . . . 619
Resynchronization of the Counter . . . 621
Configuration . . . 622
Known issues . . . 623
nmashotpconf utility cannot modify the user resynchronization window . . . 623
Other Administrative Tasks . . . 624
Using the Policy Refresh Rate Command . . . 624
Using the LoginInfo Command . . . 624
Disabling the NMAS Based Logins for LDAP . . . 628
Invoking NMAS Commands . . . 628
Setting the Delay Time for Failed Login Attempts . . . 628
Using DSTrace . . . 629
Disabling and Uninstalling the NMAS Client . . . 629
Using External Certificates with OpenText Audit . . . 629
Security Considerations . . . 629
Partner Login Methods . . . 630
Login Policies . . . 630
NMASInst . . . 631
Universal Password . . . 631
SDI Key . . . 632

25 Understanding the Certificate Server 633

OpenText Certificate Server Features . . . 633
OpenText Certificate Server Components . . . 634
OpenText Certificate Server . . . 634
NICI . . . 640
Setting Up OpenText Certificate Server . . . 641

Deciding Which Type of Certificate Authority to Use . . . 641
Creating an Organizational Certificate Authority Object . . . 642
Subordinate Certificate Authority . . . 644
Restrictions for Creating a Certificate Authority Object . . . 646
Configuring the Certificate Authority in Suite B Mode . . . 647
Creating a Server Certificate Object . . . 647
Configuring Cryptography-Enabled Applications . . . 648
Additional Components to Set Up . . . 649
Managing OpenText Certificate Server . . . 650
Certificate Authority Tasks . . . 652
Server Certificate Object Tasks . . . 661
User Certificate Tasks . . . 670
X.509 Certificate Self-Provisioning . . . 675
Using OpenText eDirectory Certificates with External Applications . . . 679
Trusted Root Object Tasks . . . 681
Certificate Revocation List (CRL) Tasks . . . 683
OpenText eDirectory Tasks . . . 690
Application Tasks . . . 691
PKI Health Check . . . 692
Public Key Cryptography Basics . . . 695
Overview . . . 696
Secure Transmissions . . . 696
Key Pairs . . . 696
Establishing Trust . . . 699
Entry Rights Needed to Perform Tasks . . . 702

26 Managing Passwords 707

Understanding Universal Password . . . 707
How Secure Is Universal Password? . . . 707
Universal Password . . . 709
Password Policies . . . 709
Password Synchronization . . . 709
Understanding Non-Reversible Password Storage . . . 710
Enabling Non-Reversible Password Storage . . . 711
Password Policies . . . 711
Deploying Universal Password . . . 711
Step 1: Identify Your Need for Universal Password . . . 712
Step 2: Make Sure Your Security Container Is Available . . . 712
Step 3: Verify That Your SDI Domain Key Servers Are Ready for Universal Password . . . 712
Step 4: Check the Tree for SDI Key Consistency . . . 714
Step 5: Enable Universal Password . . . 714
Backward Compatibility . . . 715
Password Administration . . . 715
Issues to Watch For . . . 715
Managing Passwords by Using Password Policies . . . 716
Overview of Password Policy Features . . . 717
Planning for Password Policies . . . 717
Prerequisite Tasks for Using Password Policies . . . 721
Creating Password Policies . . . 722
Assigning Password Policies to Users . . . 739
Finding Out Which Policy a User Has . . . 740
Setting A User's Password . . . 740
Universal Password Diagnostic Utility . . . 741

Troubleshooting Password Policies 742
Security Considerations 744
Importing Hash Based Passwords Into eDirectory 745

27 REST Services 747

Security Recommendations 748
Planning to Install REST Services for OpenText eDirectory 748
Configuring REST Services for OpenText eDirectory 751
Managing Data Persistence 753
Auditing with REST Services 754
Understanding REST Events 754
Modifying LDAP Password Using REST Container 755
Modifying Server Certificate Using REST Container 755
Upgrading REST Services for OpenText eDirectory 756
REST API Documentation 757

A NMAS Considerations 759

Setting Up a Security Container As a Separate Partition 759
Merging Trees with Multiple Security Containers 759
Product-Specific Operations to Perform prior to Tree Merge 760
Performing the Tree Merge 763
Product-Specific Operations to Perform after the Tree Merge 763

B OpenText eDirectory Linux Commands and Usage 765

General Utilities 765
LDAP-Specific Commands 770

C Configuring OpenSLP for OpenText eDirectory 775

Service Location Protocol 775
SLP Fundamentals 775
OpenText Service Location Providers 776
User Agents 776
Service Agents 777
Configuration Parameters 777

D How OpenText eDirectory Works with DNS 781

E Configuring GSSAPI with OpenText eDirectory 783

Concepts 783
What is Kerberos? 783
What is SASL? 784
What is GSSAPI? 784
How Does GSSAPI Work with OpenText eDirectory? 784
Prerequisites for Configuring GSSAPI 785
Assumptions on Network Characteristics 786
Adding Kerberos LDAP Extensions 786
Exporting the Trusted Root Certificate 787

Merging OpenText eDirectory Trees Configured with SASL-GSSAPI Method 787
Managing the SASL-GSSAPI Method 788
Managing a Service Principal 788
Creating a Login Sequence 789
How Does LDAP Use SASL-GSSAPI? 789
Error Messages 789
Commonly Used Terms 789

F Security Considerations 791

LDAP Binds 791
Nessus Scan Results 791

G Mapping OpenText eDirectory Events with CEF Events 793

Mapping OpenText eDirectory Events with CEF Events 793
CEF Events 797
Security Events 797
Objects Events 804
Attribute Events 806
EBA Events 808

H Troubleshooting 811

Troubleshooting SNMP 811
Troubleshooting OpenText iMonitor 815
Troubleshooting Obituaries 817
Migrating to OpenText eDirectory 820
Troubleshooting Schema 826
Troubleshooting DSRepair 827
Troubleshooting Replication 827
Troubleshooting Clone DIB Issues 828
Troubleshooting OpenText Public Key Infrastructure Services 828
Troubleshooting Utilities on Linux 834
Troubleshooting NMAS 835
Accessing HTTPSTK When Directory Service Is Not Loaded 837
Troubleshooting Data Encryption 839
The OpenText eDirectory Management Toolbox 842
Troubleshooting Issues with SASL-GSSAPI 843
Managing Error Logging in OpenText eDirectory 845
Miscellaneous 848
Troubleshooting IPV6 Issues 856
Troubleshooting EBA 856

About this Book and the Library

The *Administration Guide* describes how to manage and configure the OpenText™ eDirectory CE 24.4 (v9.3)product.

Intended Audience

This book is intended for network administrators.

Other Information in the Library

The library provides the following information resources:

Installation Guide

Describes how to install OpenText eDirectory. It is intended for network administrators.

Tuning Guide for Linux Platforms

Describes how to analyze and tune OpenText eDirectory on Linux platforms to yield superior performance in all deployments.

These guides are available at OpenText eDirectory 9.3 documentation Web site (https://www.netiq.com/documentation/edirectory-93/).

For information about the OpenText eDirectory management utility, see the OpenText™ Identity Console Administration Guide.

1 Understanding OpenText eDirectory

In simplest terms, OpenText eDirectory is a list of objects that represent network resources, such as network users, servers, printers, print queues, and applications. OpenText eDirectory is a highly scalable, high-performing, secure directory service. It can store and manage millions of objects, such as users, applications, network devices, and data. OpenText eDirectory offers a secure identity management solution that runs across multiple platforms, is internet-scalable, and extensible.

OpenText eDirectory provides centralized identity management, infrastructure, Net-wide security, and scalability to all types of applications running behind and beyond the firewall. OpenText eDirectory includes Web-based and wireless management capabilities, allowing you to access and manage the directory and users, access rights, and network resources from a Web browser and a variety of handheld devices.

OpenText eDirectory natively supports the directory standard Lightweight Directory Access Protocol (LDAP) 3 and provides support for TLS/SSL services based on the OpenSSL source code.

For more information on the OpenText eDirectory engine, see "OpenText eDirectory Process Requests" (http://support.novell.com/techcenter/articles/anp20020801.html).

Figure 1-1 shows a few of the objects as viewed in the OpenText™ Identity Console Administration Guide management utility.

Figure 1-1 *OpenText eDirectory Objects in OpenText Identity Console*



Some object classes might not be available, depending on the actual schema configured on the OpenText eDirectory server and the operating system running OpenText eDirectory.

For more information on objects, see "Object Classes and Properties" on page 23.

If you have more than one OpenText eDirectory server on the network, the directory can be replicated on multiple servers.

This chapter includes the following information:

- "Ease of Management through OpenText Identity Console" on page 20
- "Object Classes and Properties" on page 23
- "Context and Naming" on page 42

- "Schema" on page 45
- "Partitions" on page 53
- "Replicas" on page 56
- "Server Synchronization in the Replica Ring" on page 62
- "Access to Resources" on page 62
- "OpenText eDirectory Rights" on page 63

Ease of Management through OpenText Identity Console

OpenText eDirectory allows for easy, powerful, and flexible management of network resources. It also serves as a repository of user information for groupware and other applications. These applications access your directory through the industry-standard Lightweight Directory Access Protocol (LDAP).

OpenText eDirectory ease-of-management features include a powerful tree structure, an integrated management utility, and single login and authentication.

OpenText Identity Console lets you manage the directory and users, and access rights and network resources within the directory, from a Web browser and a variety of hand-held devices. The OpenText Identity Console gives you access to basic directory management tasks, and to the OpenText eDirectory management utilities you previously had to run on the OpenText eDirectory server, such as DSRepair, DSMerge, and Backup and Restore.

For more information, see the *OpenText™ Identity Console Administration Guide* (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

Powerful Tree Structure

OpenText eDirectory organizes objects in a tree structure, beginning with the top Tree object, which bears the tree's name.

Whether your OpenText eDirectory servers are running Linux or Windows, all resources can be kept in the same tree. You won't need to access a specific server or domain to create objects, grant rights, change passwords, or manage applications.

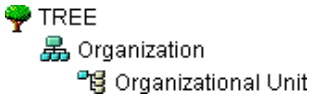
The hierarchical structure of the tree gives you great management flexibility and power. These benefits primarily result from the following two features:

- "Container Objects" on page 20
- "Inheritance" on page 21

Container Objects

Container objects allow you to manage other objects in sets, rather than individually. There are three common classes of container objects, as seen in Figure 1-2:

Figure 1-2 *Common Classes of Container Objects*



The Tree object is the top container object in the tree. It usually contains your company's Organization object.

Organization is normally the first container class under the Tree object. The Organization object is typically named after your company. Small companies keep management simple by having all other objects directly under the Organization object.

Organizational Unit objects can be created under the Organization to represent distinct geographical regions, network campuses, or individual departments. You can also create Organizational Units under other Organizational Units to further subdivide the tree.

Other classes of container objects are Country and Locality, which are typically used only in multinational networks.

The Domain object can be created under the Tree object or under Organization, Organizational Unit, Country, and Locality objects.

You can perform one task on the container object that applies to all objects within the container. Suppose you want to give a user named Amy complete management control over all objects in the Accounting container, which contains the Database application, the Bookkeepers group, the LaserPrinter printer, and the users Amy, Bill, and Bob.

To do this, navigate to the Tree View on the home page of OpenText Identity Console and select the parent tree of the **Accounting** object in the left pane. In the right pane, select **Accounting** and then click **Actions** > **Modify Trustees**. Click **Add Trustee** and add Amy as a trustee. Next, click **Assigned Rights** and select the rights you want Amy to have. Now Amy has rights to manage the Database application, the Bookkeepers group, the LaserPrinter printer, and the users Bill and Bob, in addition to herself.

Inheritance

Another powerful feature of OpenText eDirectory is rights inheritance. Inheritance means that rights flow down to all containers in the tree. This allows you to grant rights with very few rights assignments. For example, suppose you want to grant management rights to the objects shown in Figure 1-3 on page 21.

Figure 1-3 *Sample OpenText eDirectory Objects*



You could make any of the following assignments:

- If you grant a user rights to Allentown, the user can manage only objects in the Allentown container.
- If you grant a user rights to East, the user can manage objects in the East, Allentown, and Yorktown containers.
- If you grant a user rights to YourCo, the user can manage any objects in any of the containers shown.

For more information on assigning rights, see "OpenText eDirectory Rights" on page 63.

Web-Based Management Utility

OpenText Identity Console is a browser-based tool used for administering, managing, and configuring OpenText eDirectory objects. OpenText Identity Console gives you the ability to assign specific tasks or responsibilities to users and to present the user with only the tools (with the accompanying rights) necessary to perform those sets of tasks.

To run OpenText Identity Console, you will need a workstation with Microsoft Internet Explorer 6.0 SP1 or later (recommended), Mozilla 1.7 or later, or Mozilla Firefox 0.9.2 or later.

IMPORTANT: While you might be able to access OpenText Identity Console through a Web browser not listed, we do not guarantee full functionality.

You can use OpenText Identity Console to perform the following supervisory tasks:

- Configure LDAP- and XML-based access to OpenText eDirectory
- Create objects representing network users, devices, and resources
- Define templates for creating new user accounts
- Find, modify, move, and delete network objects
- Define rights and roles to delegate administrative authority
- Extend the OpenText eDirectory schema to allow custom object types and properties
- Partition and replicate the OpenText eDirectory database across multiple servers
- Run OpenText eDirectory management utilities such as DSRepair, DSMerge, and Backup and Restore

You can use OpenText Identity Console to perform other management functions. The following OpenText eDirectory tiles are available in OpenText Identity Console:

- LDAP Configuration
- Universal Password Enforcement
- Encrypted Attributes
- Authentication Management
- PKI/Certificate
- Filtered Replica Configuration Wizard
- SNMP
- WAN Traffic Manager

For more information on installing, configuring, and running Identity Console, *OpenText™ Identity Console Administration Guide* (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

Single Login and Authentication

With OpenText eDirectory, users log in to a global directory, so you don't need to manage multiple server or domain accounts for each user, and you don't need to manage trust relationships or pass-through authentication among domains.

A security feature of the directory is authentication of users. Before a user logs in, a User object must be created in the directory. The User object has certain properties, such as a name and password.

When the user logs in, OpenText eDirectory checks the password against the one stored in the directory for that user and grants access if they match.

Object Classes and Properties

The definition of each type of OpenText eDirectory object is called an object class. For instance, User and Organization are object classes. Each class of object has certain properties. A User object, for example, has First Name, Last Name, and many other properties.

The schema defines the object classes and properties, along with the rules of containment (what containers can contain which objects). Openext eDirectory ships with a base schema that you, or the applications you use, can extend. For more information about schema, see "Schema" on page 45.

Container objects contain other objects and are used to divide the tree into branches, while leaf objects represent network resources.

List of Objects

The following tables list OpenText eDirectory object classes. Added services can create new object classes in OpenText eDirectory that are not listed below.

OpenText eDirectory Container Object Classes

Identity Console Icon	Container Object (Abbreviation)	Description
	Tree	Represents the beginning of your tree. For more information, see "Tree" on page 25.
	Country (C)	Designates the countries where your network resides and organizes other directory objects within the country. For more information, see "Country" on page 28.

Identity Console Icon	Container Object (Abbreviation)	Description
	License Container (LC)	Created automatically when you install a license certificate or create a metering certificate using NetIQ Licensing Services (NLS) technology. When an NLS-enabled application is installed, it adds a License Container container object to the tree and a License Certificate leaf object to that container.
	Organization (O)	Helps you organize other objects in the directory. The Organization object is a level below the Country object (if you use the Country object). For more information, see “Organization” on page 26.
	Organizational Unit (OU)	Helps you to further organize other objects in the directory. The Organizational Unit object is a level below the Organization object. For more information, see “Organizational Unit” on page 27.
	Domain (DC)	Helps you to further organize other objects in the directory. The Domain object can be created under the Tree object or under Organization, Organizational Unit, Country, and Locality objects. For more information, see “Domain” on page 28.

OpenText eDirectory Leaf Object Classes

Identity Console Icon	Leaf Object	Description
	AFP Server	Represents an AppleTalk* Filing Protocol server that operates as a node on your OpenText eDirectory network. It usually also acts as a router to, and the AppleTalk server for, several Macintosh* computers.
	Alias	Points to the actual location of an object in the directory. Any directory object located in one place in the directory can also appear to be in another place in the directory by using an Alias. For more information, see “Alias” on page 39.
	Application	Represents a network application. Application objects simplify administrative tasks such as assigning rights, customizing login scripts, and launching applications.
	Computer	Represents a computer on the network.
	Directory Map	Refers to a directory in the file system. For more information, see “Directory Map” on page 40.
	Group	Assigns a name to a list of User objects in the directory. You can assign rights to the group instead of to each user. The rights then transfer to each user in the group. For more information, see “Group” on page 32.

Identity Console Icon	Leaf Object	Description
	License Certificate	Use with NLS technology to install product license certificates as objects in the database. License Certificate objects are added to the Licensed Product container when an NLS-aware application is installed.
	Organizational Role	Defines a position or role within an organization.
	Print Queue	Represents a network print queue.
	Print Server	Represents a network print server.
	Printer	Represents a network printing device.
	Profile	Represents a login script used by a group of users who need to share common login script commands. The users don't need to be in the same container. For more information, see "Profile" on page 41.
	Server	Represents a server running any operating system. For more information, see "Server" on page 29.
	Template	Represents standard User object properties that can be applied to new User objects.
	Unknown	Represents an object for which Identity Console has no custom icon.
	User	Represents the people who use your network. For more information, see "User" on page 31.
	Volume	Represents a physical volume on the network. For more information, see "Volume" on page 30.

Container Object Classes

- "Tree" on page 25
- "Organization" on page 26
- "Organizational Unit" on page 27
- "Country" on page 28
- "Domain" on page 28

Tree

The Tree container, formerly [Root], is created when you first install OpenText eDirectory on a server in your network. As the top-most container, it usually holds Organization objects, Country objects, or Alias objects.

What Tree Represents

Tree represents the top of your tree.

Usage

Tree is used to make universal rights assignments. Because of inheritance, any rights assignments you make to Tree as the target apply to all objects in the tree. See “OpenText eDirectory Rights” on page 63. The [Public] trustee has the Browse right and Admin has the Supervisor right to Tree by default.

Important Properties

- The Tree object has a Name property, which is the tree name you supply when installing the first server, which is shown in the tree view.
- Tree name cannot exceed 32 characters.

Organization

 An Organization container object is created when you first install OpenText eDirectory on a server in your network. As the top-most container under Tree, it usually holds Organizational Unit objects and leaf objects.

The User object named Admin is created by default in your first Organization container.

What an Organization Object Represents

Normally the Organization object represents your company, although you can create additional Organization objects under Tree. This is typically done for networks with distinct geographical districts or for companies with separate OpenText eDirectory trees that have merged.

Usage

The way you use Organization objects in your tree depends on the size and structure of your network. If the network is small, you should keep all leaf objects under one Organization object.

For larger networks, you can create Organizational Unit objects under the Organization to make resources easier to locate and manage. For example, you can create Organizational Units for each department or division in your company.

For networks with multiple sites, you should create an Organizational Unit for each site under the Organization object. That way, if you have (or plan to have) enough servers to partition the directory, you can do so logically along site boundaries.

For easy sharing of company-wide resources such as printers, volumes, or applications, create corresponding Printer, Volume, or Application objects under the Organization.

Important Properties

The most useful properties for Organization are listed below. Only the Name property is required. For a complete list of properties, select an Organization object in OpenText Identity Console. To display a description for each page of properties, click **Help**.

- Name

  Typically, the Name property is the same as your company's name. Of course, you can shorten it for simplicity. For instance, if the name of your company is Your Shoe Company, you might use YourCo.

  The Organization name becomes part of the context for all objects created under it.

- Login Script

  The Login Script property contains commands that are executed by any User objects directly under the Organization. These commands are run when a user logs in.

- Organization name can be 64 characters long.

Organizational Unit

You can create Organizational Unit (OU) container objects to subdivide the tree. Organizational Units are created with OpenText Identity Console under an Organization, Country, or another Organizational Unit.

Organizational Units can contain other Organizational Units and leaf objects such as User and Application objects.

What an Organizational Unit Object Represents

Normally the Organizational Unit object represents a department, which holds a set of objects that commonly need access to each other. A typical example is a set of Users, along with the Printers, Volumes, and Applications that those Users need.

At the highest level of Organizational Unit objects, each Organizational Unit can represent each site (separated by WAN links) in the network.

Usage

The way you use Organizational Unit objects in your tree depends on the size and structure of your network. If the network is small, you might not need any Organizational Units.

For larger networks, you can create Organizational Unit objects under the Organization to make resources easier to locate and manage. For example, you can create Organizational Units for each department or division in your company. Remember that administration is easiest when you keep User objects together in the Organizational Unit with the resources they use most frequently.

For networks with multiple sites, you can create an Organizational Unit for each site under the Organization object. That way, if you have (or plan to have) enough servers to partition the directory, you can do so logically along site boundaries.

Important Properties

The most useful properties for the Organizational Unit are listed below. Only the Name property is required. For a complete list of properties, select an Organizational Unit object in OpenText Identity Console. To display a description for each page of properties, click **Help**.

- Name

  Typically, the Name property is the same as the department name. Of course, you can shorten it for simplicity. For instance, if the name of your department is Accounts Payable, you can shorten it to AP.

  The Organizational Unit name becomes part of the context for all objects created under it.

- Login Script

  The Login Script property contains commands that are executed by any User objects directly under the Organizational Unit. These commands are run when a user logs in.

- Organizational Unit name can be 64 characters long.

Country

 You can create Country objects directly under the Tree object using OpenText Identity Console. Country objects are optional and required only for connection to certain X.500 global directories.

What a Country Object Represents

The Country object represents the political identity of its branch of the tree.

Usage

Most administrators do not create a Country object, even if the network spans countries, since the Country object only adds an unnecessary level to the tree. You can create one or many Country objects under the Tree object, depending on the multinational nature of your network. Country objects can contain only Organization objects.

If you do not create a Country object and find that you need one later, you can always modify the tree to add one.

Important Properties

- The Country object has a two-letter Name property. Country objects are named with a standard two-letter code such as US, UK, or DE.
- Country name cannot exceed 2 characters.

Domain

 You can create Domain objects directly under the Tree object using OpenText Identity Console. You can also create them under Organization, Organization Unit, Country, and Location objects.

What a Domain Object Represents

The Domain object represent DNS domain components. Domain objects let you use your Domain Name System location of services resource records (DNS SRV) to locate services in your tree.

Using Domain objects, a tree could look something like this:

`DS=Novell.DC=Provo.DC=USA`

In this example, all subcontainers are domains. You can also use Domain objects in a mixed tree, such as:

`DC=Novell.O=Provo.C=USA`

Or

`OU=Novell.DC=Provo.C=USA`

Usually, the topmost Domain is the overall Tree, with subdomains under Tree. For example, machine1.novell.com could be represented by `DC=machine1.DC=novell.DC=com` in a tree representation. Domains give you a more generic way to set up an OpenText eDirectory tree. If all containers and subcontainers are DC objects, users do not need to remember C, O, or OUs when searching for objects.

Usage

Domain name can be 64 characters long.

Leaf Object Classes

- "Server" on page 29
- "Volume" on page 30
- "User" on page 31
- "Group" on page 32
- "Nested Groups" on page 35
- "Alias" on page 39
- "Directory Map" on page 40
- "Profile" on page 41

Server

 A Server object is automatically created in the tree whenever you install OpenText eDirectory on a server. The object class can be any server running OpenText eDirectory.

What a Server Object Represents

The Server object represents a server running OpenText eDirectory or a bindery-based server.

Usage

The Server object serves as a reference point for replication operations. A Server object that represents a bindery-based server allows you to manage the server's volumes with OpenText Identity Console.

Important Properties

The Server object has a Network Address property, among others. The Network Address property displays the protocol and address number for the server. This is useful for troubleshooting at the packet level

For a complete list of properties, select a Server object in OpenText Identity Console. To display a description for each page of properties, click **Help**.

Volume

 When you create a physical volume on a server, a Volume object is automatically created in the tree. By default, the name of the Volume object is the server's name with an underscore and the physical volume's name appended (for example, Y0SERVER_SYS).

Linux file system partitions cannot be managed using Volume objects. Volume objects are supported only on OES Linux.

What a Volume Object Represents

A Volume object represents a physical volume on a server, whether it is a writable disk, a CD, or other storage medium. The Volume object in OpenText eDirectory does not contain information about the files and directories on that volume, although you can access that information through OpenText Identity Console. File and directory information is retained in the file system itself.

Usage

In OpenText Identity Console, click the **Volume** icon to manage files and directories on that volume. Identity Console provides information about the volume's free disk space, directory entry space, and compression statistics.

Important Properties

In addition to the required Name and Host Server properties, there are other important Volume properties.

- Name

  This is the name of the Volume object in the tree. By default, this name is derived from the name of the physical volume, though you can change the object name.

- Host Server

  This is the server that the volume resides on.

- Version

  This is the OpenText eDirectory version of the server hosting the volume.

User

A User object is required for logging in. When you install the first server into a tree, a User object named Admin is created. Log in as Admin the first time.

You can use the following methods to create or import User objects:

- OpenText Identity Console

  For more information about OpenText Identity Console see the *OpenText™ Identity Console Administration Guide* (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

- Batches from database files

  For more information on using batch files, see "Designing the OpenText eDirectory Tree" on page 76.

What a User Object Represents

A User object represents a person who uses the network.

Usage

You should create User objects for all users who need to use the network. Although you can manage User objects individually, you can save time by

- Using Template objects to set default properties for most User objects. The Template applies automatically to new Users you create (not to already existing ones).
- Creating Group objects to manage sets of Users.
- Assigning rights using the container objects as trustees when you want that assignment to apply to all User objects in the container.
- Selecting multiple User objects by using Shift+click or Ctrl+click. When you do, you can change property values for all selected User objects.

Important Properties

User objects have over 80 properties. For a complete list of properties, select a User object in OpenText Identity Console. To display a description for each page of properties, click **Help**.

The Login Name and Last Name properties are required. These and some of the most useful properties are listed below.

- Account Expiration Date lets you limit the life of a user account. After the expiration date, the account is locked so the user cannot log in.
- Account Disabled has a system-generated value that indicates a lock on the account so the user cannot log in. The lock might occur if the account has expired or because the user has given too many incorrect passwords in succession.
- Force Periodic Password Changes lets you enhance security by requiring the user to change passwords after a specified interval.
- Group Memberships lists all the Group objects that include the User as a member.

- Last Login is a system-generated property that lists the date and time that the user last logged in.
- Last Name, although required, is not used directly by OpenText eDirectory. Applications that take advantage of the OpenText eDirectory name base can use this property, along with other identification properties such as Given Name, Title, Location, and Fax Number.
- Limit Concurrent Connections lets you set the maximum number of sessions a user can have on the network at any given time.
- Login Name is the name that is shown in the OpenText Identity Console by the User icon. It is also the name supplied by the user when logging in.

  OpenText eDirectory does not require that login names be unique throughout the network, only in each container. However, you might want to keep login names unique across the company to simplify administration.

  Typically, login names are a combination of first and last names, such as STEVEJ or SJONES for Steve Jones.
- Login Script lets you create specific login commands for a User object. When a user logs in, the container login script runs first. Then a profile login script runs if the User object has been added to the membership list of a Profile object. Finally, the user login script runs (if one exists).

  You should put most of the login commands in container login scripts to save administrative time. The user login script can be edited to manage unique exceptions to common needs.
- Login Time Restrictions lets you set times and days when the user can log in.
- Network Addresses contains system-generated values that list all the IPX™ and/or IP addresses that the user is logged in from. These values are useful for troubleshooting network problems at the packet level.
- Require a Password lets you control whether the user must use a password. Other related properties let you set common password constraints such as password length.
- Rights to Files and Directories lists all rights assignments made for this user to the file system. Using OpenText Identity Console, you can also check a user's effective rights to files and directories, which include those inherited from other objects.

Group

 You can create Group objects to help you manage sets of User objects.

What a Group Object Represents

A Group object represents a set of User objects.

Usage

Container objects let you manage all User objects in that container, and Group objects are for subsets within a container or in multiple containers.

Group objects have two main purposes:

- They allow you to grant rights to a number of User objects at once.
- They allow you to specify login script commands using the `IF MEMBER OF` syntax.

Static Groups

Static groups identify the member objects explicitly. Each member is assigned to the group explicitly.

These groups provide a static list of members, as well as referential integrity between the members list of the group and the members of attributes on an object. Group membership is managed explicitly through the member attribute.

Dynamic Groups

Dynamic groups use an LDAP URL to define a set of rules which, when matched by OpenText eDirectory User objects, define the members of the group. Dynamic group members share a common set of attributes as defined by the search filter specified in the URL. For more information on the LDAP URL format, see RFC 2255 (http://www.ietf.org/rfc/rfc2255.txt).

Dynamic groups let you specify the criteria to be used for evaluating membership in a group. The actual members of the group are dynamically evaluated by OpenText eDirectory, which lets you define the group members in terms of a logical grouping and lets OpenText eDirectory automatically add and remove group members. This solution is more scalable, reduces administrative costs, and can supplement normal groups in LDAP to provide increased flexibility.

OpenText eDirectory lets you create a dynamic group when you want to automatically group users based on any attribute, or when you want to apply ACLs to specific groups that contain matching distinguished names (DNs). For example, you can create a group that automatically includes any DN that contains the attribute Department=Marketing. If you apply a search filter for Department=Marketing, the search returns a group including all DNs containing the attribute Department=Marketing. You can then define a dynamic group from the search results based on this filter. Any User added to the directory who matches the Department=Marketing criteria is automatically added to the group. Any User whose Department is changed to another value (or who is removed from the directory) is automatically removed from the group.

Dynamic groups are created in OpenText eDirectory by creating an object of type objectclass=dynamicGroup. A static Group object can be converted into a dynamic group by associating an auxiliary class, dynamicGroupAux, to the Group object. The dynamic group has the memberQueryURL attribute associated with it.

A dgIdentity attribute can be set on the Dynamic Group object to the distinguished name of an entry, whose credentials and rights should be used to expand the dynamic members of the group.

The groups are managed using the memberQueryURL. A typical memberQueryURL has a base DN, a scope, a filter, and an optional extension. The base DN specifies the search base. Scope specifies the levels below the base to search, and filter is the search filter based on which entries are selected from within the specified scope.

NOTE: To address exceptions to the listing created by the memberQueryURL, dynamic groups also allow for explicit inclusion and exclusion of users.

Dynamic groups can be created and managed through OpenText Identity Console You can access group management tasks by clicking the **Group Management** tile.

You can also use LDAP commands to manage such groups. The most useful properties associated with dynamic groups are dgIdentity and memberQueryURL.

Important Properties

The most useful properties of the Group object are Members and Rights to Files and Directories. For a complete list of properties, select a Group object in Identity Console. To display a description for each page of properties, click **Help**.

- Creating a Group
- Deleting Groups
- Modifying Groups
- Adding or Modifying Groups
- Searching for Groups

Upgrading Dynamic Groups on Pre-OpenText eDirectory 8.6.1 Databases

Dynamic group functionality requires some internal values stored on the Dynamic Group objects, which are created either when a dynamic group is locally created or received as a part of synchronization.

Although older servers can hold dynamic groups, they are unable to generate these values, because dynamic groups were introduced in OpenText eDirectory 8.6.1.

In OpenText eDirectory 8.6.2, automatic upgrade of the Dynamic Group objects in a pre-8.6.1 database to match a OpenText eDirectory 8.6.1 database was implemented.

Support for Additional Syntaxes in memberQueryURL

The memberQueryURL attribute can hold a search filter that the OpenText eDirectory server uses to compute the members of a dynamic group.

In OpenText eDirectory 8.6.1, the syntaxes of attributes used in the filter were restricted only to the following basic string types:

- SYN_CE_STRING
- SYN_CI_STRING
- SYN_PR_STRING
- SYN_NU_STRING
- SYN_CLASS_NAME
- SYN_TEL_NUMBER
- SYN_INTEGER
- SYN_COUNTER
- SYN_TIME
- SYN_INTERVAL
- SYN_BOOLEAN
- SYN_DIST_NAME
- SYN_PO_ADDRESS
- SYN_CI_LIST

- SYN_FAX_NUMBER
- SYN_EMAIL_ADDRESS

From OpenText eDirectory 8.7.3 onwards, the following additional attribute syntaxes are supported in a memberQueryURL value:

- SYN_PATH
- SYN_TIMESTAMP
- SYN_TYPED_NAME

In both OpenTexte Directory 8.6.1 and OpenText eDirectory 8.7.x, binary syntaxes like SYN_OCTET_STRING and SYN_NET_ADDRESS are not supported in the memberQueryURL search filters.

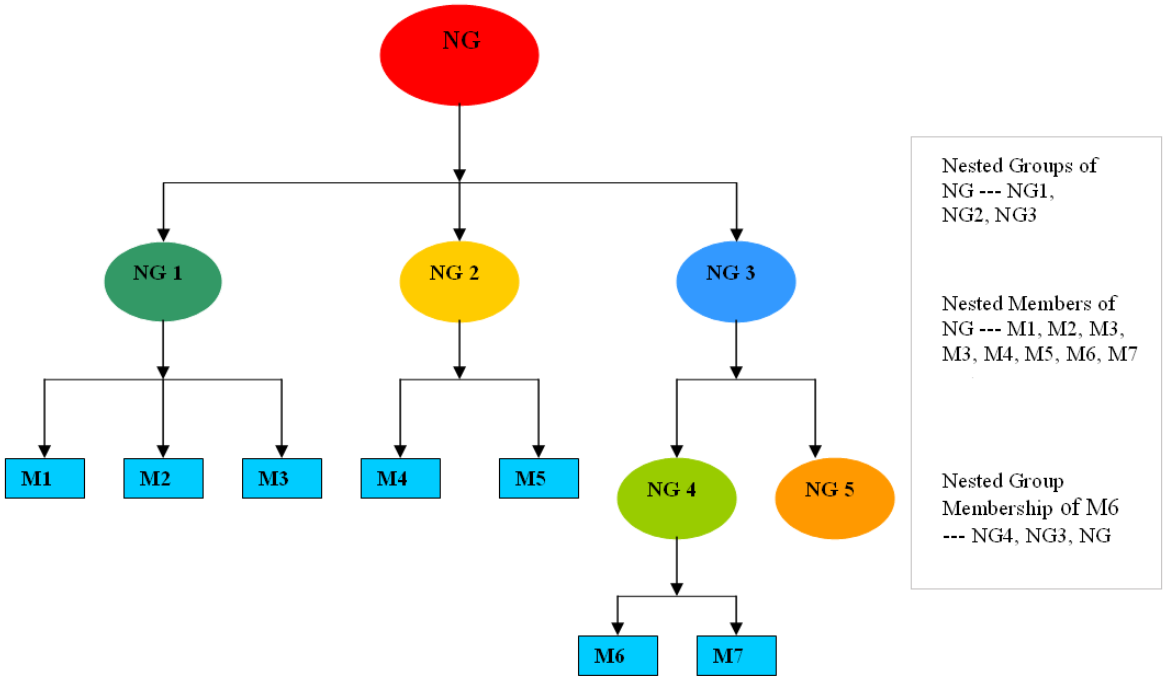
For more information, see "How to Manage and Use Dynamic Groups in OpenText eDirectory" (http://support.novell.com/techcenter/articles/ana20020405.html).

Nested Groups

Nested groups allow grouping of groups and provide a more structured form of grouping. The groupMember attribute specifies the nested groups whose members become nested members of the containing nested group object. Group objects are specified statically in the groupMember attribute. The group containing other groups is referred to as the containing group, and the groups that are part of this group are referred to as contained groups. OpenText eDirectory supports nesting of both static and dynamic groups. Nesting can have multiple levels up to 200.

IMPORTANT: Nesting is supported within the local server only. If a contained group is not found on the local server, its members are not listed as the nested members of the containing group.

Figure 1-4 *Nested Groups*



You can use OpenText Identity Console or LDAP tools to create the nested groups.

- "Creating Nested Groups by Using LDAP Tools" on page 36
- "Creating Nested Groups by Using OpenText Identity Console" on page 36

Creating Nested Groups by Using LDAP Tools

You can use LDAP tools to create the nested groups. A new auxiliary class, nestedGroupAux, along with the structural class Group represents a nested group. This auxiliary class can be added to the existing static group object to convert it into a nested group.

Both the contained and the containing group should be nested group objects. Only when the contained group is a nested group, it can populate the groupMembership attribute (groupMembership attribute not a part of static group) on it to specify the containing group. If the contained groups are not of the same type as the containing group, only the static members of the contained group are listed as nested members.

You can use LDIF files and LDAP tools to manage such groups. The most useful properties associated with nested groups are groupMember and nestedConfig.

Creating Nested Groups by Using OpenText Identity Console

You can use OpenText Identity Console tile s to create a nested group or to change a static group to a nested group in order to associate it with another group.

1. Log in to OpenText Identity Console with administrator credentials and select **Groups Management** > **Create Group** from the left panel to create a static group.
2. Select **Groups** > **Create Group** from the left panel, select the **Enable Nested Group** check box to create a nested group, then click **Save**.

3 Select **Groups** > **Modify Group**, select the **Nested Group** check box to modify the nested group, then click **Save** and click **OK**.

4 To modify the nested group, select **Groups > Modify Group**, then browse to and select the desired group to be modified.

5 To associate a static group, select the **Nested** tab **> Group Member** tab, then browse to and select the static group.

6 Click **Save**, then click **OK** to convert the static group to the nested group.

The static group that you converted to a nested group is now a member of the static group.

To verify the membership details of static group, select **Groups > Modify Group**, then select the static group. Select the **Nested > Group Membership** tab to verify the static group's membership information. Click + to add a member to the group and 🗑 to exclude or remove the member assigned to the group.

Nested Group Properties

- member

  A nested group's member attribute lists all direct members of the nested group and expands the nested member groups in to their members.

  NOTE: Enumerating nested group membership is done only when there is an assertion on the member attribute. Therefore, if there are many nested member groups containing many users, enumerating the total group membership could generate load on OpenText eDirectory.

- groupMember

  A nested group's `groupMember` attribute does not expand a nested member group in to its members and only shows the nested group names.

- Group Membership

  groupMembership specifies the group that this object (generally a user object) belongs to. This attribute is associated with the nestedGroupAux class, and it holds the DN of the nested group of which this group is a group member. When associated with a group object, it indicates the nested group of which this group is a member (specifically a groupMember). Similar to member and groupMember, groupMembership lists all the nested groups of which this group has a groupMembership via a nested relationship. The nestedConfig also applies to the groupMembership attribute. For non-group member objects, the nestedConfig of individual groups is used.

Nested Group Operations

1. One group can be a member of another group via the groupMember attribute. Both groups, contained as well containing, must have the nested group auxiliary class associated with the group object.

   ```
   dn: cn=finance,o=nov

   objectclass: group

   objectclass: nestedGroupAux
   ```

```
groupMember: cn=accounts,o=nov

member: cn=jim,o=nov


dn: cn=accounts,o=nov

objectclass: group

objectclass: nestedGroupAux

member: cn=allen,o=nov

member: cn=ESui,o=nov

member: cn=YLi,o=nov
```

2. Reading the member attribute of a nested group also causes the members of the contained group to be returned if both the contained and the containing group are present locally on the server:

```
dn: cn=finance,o=nov
member: cn=jim,o=nov
member: cn=allen,o=nov
member: cn=ESui,o=nov
member: cn=YLi,o=nov
```

   The same holds true for the groupMember attribute.

3. The reciprocal attribute to the member attribute is groupMembership. This implies that the `cn=allen,o=nov` user object needs to possess the groupMembership attribute populated with the `cn=accounts,o=nov` group DN. The groupMembership of the `cn=accounts,o=nov` group needs to be populated with `cn=finance,o=nov`. On reading the groupMembership attribute of the `cn=allen,o=nov` user object, both the groups are returned.

```
dn: cn=allen,o=nov
groupMembership: cn=accounts,o=nov
groupMembership: cn=finance,o=nov
```

4. The ACLs can be assigned to a nested group and all the objects that are members of the nested group will acquire the rights. In the assigned rights field, an additional nested ACL bit (0x80000000) needs to be set in addition to the rights being assigned.

```
dn: cn=finance,o=nov
groupMember: cn=accounts,o=nov


dn: cn=accounts,o=nov
member: cn=allen,o=nov


dn: ou=MyCo,o=nov
objectclass: Organizational Unit
ACL: 2147483650#entry#cn=finance,o=nov#[All Attributes Rights]
```

The rights value – 2147483650 (0x80000002) has nested ACL (0x80000000) and read rights bit (0x00000002) set. So, the user object `cn=allen,o=nov` has been granted read rights on all attributes of the MyCo object via the nested group `cn=finance,o=nov`.

5. Applications can use filter assertions on the member, groupMember, and groupMembership attributes. In the above example, an assertion of `member=cn=allen,o=nov` would return the following:

   ```
   dn: cn=accounts,o=nov
   dn: cn=finance,o=nov
   ```

   An assertion of `groupMembership=cn=finance,o=nov` would return the following objects:

   ```
   dn: cn=allen,o=nov
   dn: cn=jim,o=nov
   dn: cn=ESui,o=nov
   dn: cn=YLi,o=nov
   dn: cn=accounts,o=nov
   ```

NOTE: There is no limit on the levels of nesting in any of the above cases. Loop detection in nested groups is done while any of the above mentioned attributes are read.

Limitations

- Nested relationships do not span beyond the local server. The objects, users, and groups involved need to be locally present on the server.
- No duplicate elimination is done in membership listing.
- Nested ACLs as well as the nesting semantics are not supported on older OpenText eDirectory servers (version 8.8 SP1 and earlier).

Alias

You can create an Alias object that points to another object in the tree. An Alias object gives a user a local name for an object that lies outside their container.

When you rename a container, you have the option of creating an Alias in the former container's place that points to the new name. Workstations and login script commands that reference objects in the container can still access the objects without having the container name updated.

What an Alias Object Represents

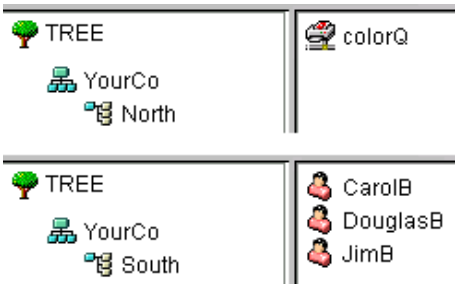
An Alias object represents another object, which can be a container, User object, or any other object in the tree. An Alias object does not carry trustee rights of its own. Any trustee authority you grant to the Alias object applies to the object it represents. The Alias can be a target of a trustee assignment, however.

Usage

Create an Alias object to make name resolution easier. Because object naming is simplest for objects in the current context, you should create Alias objects there that point to any resources outside the current context.

For example, suppose users log in and establish a current context in the South container as shown in Figure 1-5, but need access to the Print Queue object named ColorQ in the North container.

Figure 1-5 *Sample Containers*



You can create an Alias object in the South container, as shown in Figure 1-6.

Figure 1-6 *Alias Object in OpenText eDirectory Container*



The Alias object points to the original ColorQ object, so setting up printing for the users involves a local object.

Important Properties

Alias objects have an Aliased Object property, which associates the Alias object with the original object.

Directory Map

The Directory Map object is a pointer to a path in the server file system. It allows you to make simpler references to directories.

If your network has no volumes, you cannot create Directory Map objects.

What a Directory Map Object Represents

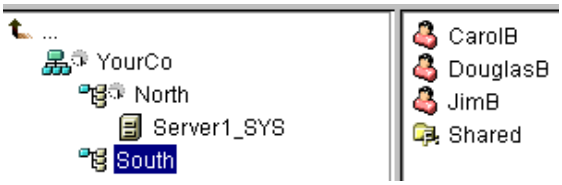
A Directory Map object represents a directory on a volume. An Alias object, on the other hand, represents an object.

Usage

Create a Directory Map object to make drive mapping simpler, particularly in login scripts. Using a Directory Map object allows you to reduce complex file system paths to a single name.

Also, when you change the location of a file, you don't need to change login scripts and batch files to reference the new location. You only need to edit the Directory Map object. For example, suppose you were editing the login script for the container South, shown in Figure 1-7.

Figure 1-7 *Sample OpenText eDirectory Container*



A command mapping drives to the Shared directory on volume `sys:` would look like the following:

```
MAP N:=sys.North.:Shared
```

If you created the Shared Directory Map object, the map command would be much simpler:

```
MAP N:=Shared
```

Important Properties

The Directory Map object has the following properties:

- Name

  Identifies the object in the directory (for example, Shared) and is used in MAP commands.

- Volume

  Contains the name of the Volume object that the Directory Map object references, such as Sys.North.YourCo.

- Path

  Specifies the directory as a path from the root of the volume, such as `public\winnt\nls\english`.

Profile

Profile objects help you manage login scripts.

What a Profile Object Represents

A Profile object represents a login script that runs after the container login script and before the user login script.

Usage

Create a Profile object if you want login script commands to run for only selected users. The User objects can exist in the same container or be in different containers. After you have created the Profile object, you add the commands to its Login Script property. Then make the User objects trustees of the Profile object and add the Profile object to their Profile Membership property.

Important Properties

The Profile object has two important properties:

- Login Script

  Contains the commands you want to run for users of the Profile.

- Rights to Files and Directories

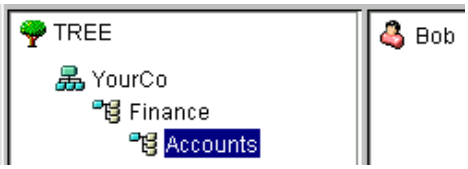
  If you have INCLUDE statements in the login script, you need to give the Profile object rights to the files included with the Rights to Files and Directories property.

Context and Naming

The context of an object is its position in the tree. It is nearly equivalent to a DNS domain.

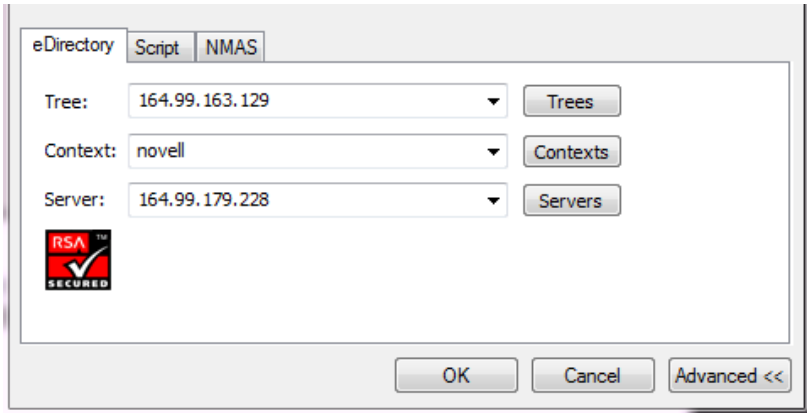
You can see in Figure 1-8 that User Bob is in Organizational Unit Accounts, which is in Organizational Unit Finance, which is in Organization YourCo.

Figure 1-8 *Sample OpenText eDirectory Container*



Sometimes, however, you need to express the context of an object in an OpenText eDirectory utility. For example, you could be setting up Bob's workstation and need to supply a name context, as shown in Figure 1-9.

Figure 1-9 *Novell Client NDS Page*



The context is specified as a list of containers separated by periods, between the object in question and the top of the Tree.

Distinguished Name

The distinguished name of an object is its object name with the context appended. For example, the complete name of User object Bob is Bob.Accounts.Finance.YourCo.

Typeful Name

Sometimes typeful names are displayed in OpenText eDirectory utilities. Typeful names include the object type abbreviations listed in the following table:

Object Class	Type	Abbreviation
All leaf object classes	Common Name	CN
Organization	Organization	O
Organizational Unit	Organizational Unit	OU
Country	Country	C
Locality	Locality or State/Province	L or S

In creating a typeful name, OpenText eDirectory uses the type abbreviation, an equal sign, and the object's name. For instance, Bob's partial typeful name is CN=Bob. Bob's complete typeful name is CN=Bob.OU=Accounts.OU=Finance.O=YourCo. You can use typeful names interchangeably with typeless names in OpenText eDirectory utilities.

Name Resolution

The process OpenText eDirectory uses to find an object's location in the directory tree is called *name resolution*. When you use object names in OpenText eDirectory utilities, it resolves the names relative to either the current context or the top of the tree.

Current Workstation Context

Workstations have a context set when the networking software runs. This context relatively identifies the location of the workstation in the network. For example, Bob's workstation would be set to the current context as follows:

```
Accounts.Finance.YourCo
```

Current context is a key to understanding the use of leading periods, relative naming, and trailing periods, discussed in the following sections.

Leading Period

Use a leading period to resolve the name from the top of the tree, no matter where the current context is set. In the example below, the leading period tells the CX (Change Context) utility to resolve the name relative to the top of the tree.

```
CX .Finance.YourCo
```

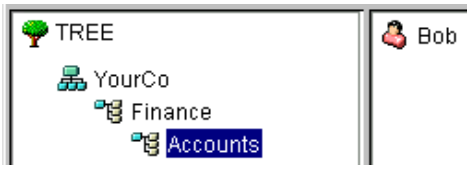
OpenText eDirectory interprets the command as “Change the context to the Finance container, which is in the YourCo container, resolved from the top of the tree.”

Relative Naming

Relative naming means that names are resolved relative to the workstation’s current context, rather than from the top of the tree. Relative naming never involves a leading period, since a leading period indicates resolution from the top of the tree.

Suppose a workstation’s current context is set to Finance. See Figure 1-10.

Figure 1-10 *Sample OpenText eDirectory Container*



The relative object name of Bob is

```
Bob.Accounts
```

OpenText eDirectory interprets the name as “Bob, which is in Accounts, resolved from the current context, which is Finance.”

Trailing Periods

Trailing periods can be used only in relative naming. Therefore, you can’t use both a leading period and a trailing period. A trailing period changes the container that OpenText eDirectory resolves the name from.

Each trailing period changes the resolution point one container toward the top of the tree. For example, suppose you want to change your workstation’s current context from Timmins to Allentown in the example in Figure 1-11.

Figure 1-11 *Sample OpenText eDirectory Container*



The proper CX command uses relative naming with trailing periods:

```
CX Allentown.East..
```

OpenText eDirectory interprets the command as “Change the context to Allentown, which is in East, resolved from two containers up the tree from the current context.”

Similarly, if Bob is in the Allentown container and your workstation’s current context is Timmins, then Bob’s relative name would be

```
Bob.Allentown.East..
```

Context and Naming on Linux

When Linux user accounts are migrated to OpenText eDirectory, the OpenText eDirectory context is not used to name users.

Schema

Schema defines the types of objects that can be created in your tree (such as Users, Printers, and Groups) and what information is required or optional at the time the object is created. Every object has a defined schema class for that type of object.

The schema that originally shipped with the product is called the base schema. After the base schema has been modified in any way, such as adding a new class or a new attribute, then it is considered the extended schema.

You aren't required to extend the schema, but you have the ability to do so. The Schema role in OpenText Identity Console lets you extend the schema to meet organizational needs. For example, if your organization requires special footwear for employees and you need to keep track of employee shoe sizes, you might want to create a new attribute called `Shoe Size` and add the attribute to an auxiliary class. You can then use that auxiliary class to extend User objects as needed. For more information about creating auxiliary classes, see “Creating an Auxiliary Class” on page 134.

For more information about working with the OpenText eDirectory schema, see Chapter 5, “Managing the Schema,” on page 131.

Schema Management

The Schema role in OpenText Identity Console lets users who have the Supervisor rights to a tree customize the schema of that tree. The Schema role, and its associated tasks, is available on the Schema Management page in OpenText Identity Console.

Use the Schema role to

- View a list of all classes and attributes in the schema.
- View information on an attribute such as its syntax and flags.
- Extend the schema by adding a class or an attribute to the existing schema.
- Create a class by naming it and specifying attributes, flags, containers that it can be added to, and parent classes that it can inherit attributes from.
- Create an attribute by naming it and specifying its syntax and flags.

- Add an optional attribute to an existing class.
- Delete a class or attribute that is not used or that is obsolete.

Schema Classes, Attributes, and Syntaxes

- "Classes" on page 46
- "Attributes" on page 46
- "Syntaxes" on page 47
- "High-Valued Attributes (HVA)" on page 50

Classes

A class is like a template for a directory object. A directory object is a class that has been filled in with data. In other words:

CLASS + DATA = DIRECTORY OBJECT

Each class has a class name, an inheritance class (unless it is at the top of the class hierarchy), class flags, and a group of attributes. Classes are named like directory objects (User, Printer, Queue, Server, etc.), yet they are just structure, with no content.

An inheritance class is a class that is a starting point for defining other object classes. All of the attributes of the inheritance class are inherited by the classes that come below it in the class hierarchy.

A class hierarchy shows how a class is associated with its parent classes. This is a way of associating similar classes and allowing attributes to be inherited. It also defines the types of containers the class is valid in.

When creating a new class, you can use the class hierarchy and the additional attributes available to customize each class. You can specify an inheritance class (which allows the new class to inherit all of the attributes and flags of a class higher in the hierarchy) and then customize the new class by selecting one or more attributes to add to those that were inherited. The additional attributes can be selected as mandatory, naming, or optional attributes.

You can also modify existing classes by adding optional attributes.

Attributes

Attributes are the data fields in the OpenText eDirectory database. For example, if a class is like a form, then an attribute is one field on the form. When an attribute is created, it is named (*surname* or *employee number*) and given a syntax type (*string* or *number*). From then on, it is available in the attribute lists in Schema Manager.

NOTE: Due to a replication issue, attributes in OpenText eDirectory other than the stream attribute type cannot contain values larger than 60 KB or 30,000 characters. If a user or application sets the value of a string or binary attribute and exceeds that limit, OpenText eDirectory returns a -649 error indicating that the value is too long.

Syntaxes

There are several syntax options to choose from. These are used to specify the type of data entered for each attribute. The syntax can be specified only when an attribute is created. You cannot modify it later. Available syntaxes include the following:

- Back Link

  Used to keep track of other servers referring to an object. It is used for internal OpenText eDirectory management purposes.

- Boolean

  Used by attributes whose values are True (represented as 1) or False (represented as 0). The single-valued flag is set for this syntax type.

- Case Exact String

  Used by attributes whose values are Unicode strings that are case sensitive in comparison operations. Two Case Exact Strings match when they are of the same length and their corresponding characters, including case, are identical.

- Case Ignore List

  Used by attributes whose values are ordered sequences of Unicode strings that are not case sensitive in comparisons operations. Two Case Ignore Lists match if the number of strings in each is the same and all corresponding strings match (that is, they are the same length and their corresponding characters are identical).

- Case Ignore String

  Used by attributes whose values are Unicode strings that are not case sensitive in comparison operations. Two Case Ignore Strings match when they are of the same length and their corresponding characters are identical in all respects except that of case.

- Class Name

  Used by attributes whose values are object class names. Two Class Names match when they are of the same length and their corresponding characters are identical in all respects except that of case.

- Counter

  Used by attributes whose values are incrementally modified numeric signed integers. Any attribute defined using Counter is a single-valued attribute. This syntax differs from Integer in that any value added to an attribute of this syntax is arithmetically added to the total, and any value deleted is arithmetically subtracted from the total.

- Distinguished Name

  Used by attributes whose values are the names of objects in the OpenText eDirectory tree. Distinguished Names (DN) are not case sensitive, even if one of the naming attributes is case sensitive.

- EMail Address

  Used by attributes whose values are strings of binary information. OpenText eDirectory makes no assumption about the internal structure of the content of this syntax.

- Facsimile Telephone Number

Specifies a string that complies with the E.123 standard for storing international telephone numbers and an optional bit string formatted according to recommendation T.20. Facsimile Telephone Number values match when they are of the same length and their corresponding characters are identical, except that all spaces and hyphen characters are ignored during comparison.

- Hold

  Used by attributes that are accounting quantities, whose values are signed integers. This syntax is an accounting quantity (which is an amount tentatively held against a subject's credit limit, pending completion of a transaction). The hold amount is treated similarly to the Counter syntax, with new values added to or subtracted from the base total. If the evaluated hold amount goes to 0, the Hold record is deleted.

- Integer

  Used by attributes represented as signed numeric values. Two Integer values match if they are identical. The comparison for ordering uses signed integer rules.

- Integer 64

  Used by attributes represented as 64-bit integer values. Integer 64 attributes support the Microsoft Large Integer Syntax and can be used to store large-integer values or dates previous to 1970 or beyond 2038.

  NOTE: OpenText eDirectory uses its existing syntax and 32-bit values for internal timestamps.

- Interval

  Used by attributes whose values are signed numeric integers and represent intervals of time. The Interval syntax uses the same representation as the Integer syntax. The Interval value is the number of seconds in a time interval.

- Net Address

  Represents a network layer address in the server environment. The address is in binary format. For two values of Net Address to match, the type, length, and value of the address must match.

- Numeric String

  Used by attributes whose values are numerical strings as defined in the CCITT X.208 definition of Numeric String. For two Numeric Strings to match, the strings must be the same length and their corresponding characters must be identical. Digits (0...9) and space characters are the only valid characters in the numeric string character set.

- Object ACL

  Used by attributes whose values represent Access Control List (ACL) entries. An Object ACL value can protect either an object or an attribute.

- Octet List

  Describes an ordered sequence of strings of binary information or Octet String. An Octet List matches a stored list if it is a subset of the stored list. For two Octet Lists to match, they must be the same length, and the corresponding bit sequence (octet) must be identical.

- Octet String

  Used by attributes whose values are strings of binary information not interpreted by OpenText eDirectory. These octet strings are non-Unicode strings. For two octet strings to match, they must be the same length, and the corresponding bit sequence (octet) must be identical.

- Path

Attributes that represent a file system path contain all the information to locate a file on a server. Two paths match when they are of the same length and their corresponding characters, including case, are identical.

- Postal Address

  Used by attributes whose values are Unicode strings of postal addresses. An attribute value for Postal Address is typically composed of selected attributes from the MHS Unformatted Postal O/R Address Specification version 1 according to recommendation F.401. The value is limited to six lines of 30 characters each, including a postal country name. Two postal addresses match if the number of strings in each is the same and all corresponding strings match (that is, they are the same length and their corresponding characters are identical).

- Printable String

  Used by attributes whose values are printable strings, as defined in CCITT X.208. The printable character set consists of the following:

  - Uppercase and lowercase alphabetic characters
  - Digits (0...9)
  - Space character
  - Apostrophe (’)
  - Left and right parentheses ( )
  - Plus sign (+)
  - Comma (,)
  - Hyphen (-)
  - Period (.)
  - Forward slash (/)
  - Colon (:)
  - Equals sign (=)
  - Question mark (?)

  Two printable strings are equal when they are the same length and their corresponding characters are the same. Case is significant.

- Replica Pointer

  Used by attributes whose values represent partition replicas. A partition of an OpenText eDirectory tree can have replicas on different servers. The syntax has six components:

  - Server Name
  - Replica Type (master, secondary, read-only, subordinate reference)
  - Replica Number
  - Replica Root ID
  - Number of Address
  - Address Record

- Stream

Represents arbitrary binary information. The Stream syntax provides a way to make an OpenText eDirectory attribute out of a file on a file server. Login scripts and other stream attributes use this syntax. The data stored in a stream file has no syntax enforcement of any kind. It is completely arbitrary data, defined by the application that created and uses it.

- Telephone Number

  Used by attributes whose values are telephone numbers. Two telephone numbers match when they are of the same length and their corresponding characters are identical, except that all spaces and hyphen characters are ignored during comparison.

- Time

  Used by attributes whose values are unsigned integers and represent time expressed in seconds.

- Timestamp

  Used by attributes whose values mark the time when a particular event occurred. When a significant event occurs, an OpenText eDirectory server mints a new Timestamp value and associates the value with the event. Every Timestamp value is unique within an OpenText eDirectory partition. This provides a total ordering of events occurring on all servers holding replicas of a partition.

- Typed Name

  Used by attributes whose values represent a level and an interval associated with an object. This syntax names an OpenText eDirectory object and attaches two numeric values to it:

  - Level of the attribute indicative of its priority
  - Interval representing the number of seconds between certain events or the frequency of the reference

- Unknown

  Used by attributes whose attribute definition has been deleted from the schema. This syntax represents strings of binary information.

High-Valued Attributes (HVA)

HVA`Config` attribute is configurable attribute. The deployment process is simple and time efficient. By default, HVA`Config` monitors `Dir-XMLEntitlementResults` and `oidpInstanceData`. If the customer uses custom values for HVA`Config`, then these two attributes must be configured again with appropriate values. Custom HVA`Config` attribute can be configured using an `ldif` file. HVA`Config` can monitor all attribute types except STREAM. All the attribute related values provided to HVA`Config` must be provided in JSON format.

The 4 mandatory JSON keys used in configuring HVA are as follows:

1. AttributeName- Name of the attribute to be monitored.
2. Type - This key refers to the attribute type that is monitored. For example, 1 = count-based attribute, 2 = size-based attribute.
3. Limit- The upper limit after which high value alert/logging starts.
4. Interval- This key is used in case of count-based attributes. It shows an alert every interval count after the attribute exceeds limit.

There are various ways to add / modify HVAConfig attribute as described below.

1. Adding HVAConfig attribute to NCP server object using ldif file.

   Sample ldif file is shown below.

   ```
   version:1
   # define attributes
   dn: cn=servername,o=server
   changetype: modify
   add: HVAConfig
   HVAConfig:
   [{"AttributeName":"oidpInstanceData","Type":2,"Limit":1,"Interval":0}]
   ```

   where dn: is ncp server object DN.

   Use `ldapmodify` to upload the ldif file as shown below.

   ```
   ldapmodify -h 10.xx.xx.xx -D cn=admin,o=novell -w password -f
   hvaPolicy.ldif
   ```

2. Adding HVAConfig attribute from OpenText Identity Console.
   1. Go to Directory Administration from Roles and `Tasks -> Modify Objects-> select NCP server Object`. Click OK.
   2. Select **"Others"** tab in modify page from **"Unvalued Attributes"** and select HVAConfig.
   3. Enter the desired attribute to set limit for in JSON format.

      Example:`[{"AttributeName":"oidpInstanceData","Type":2,"Limit":1,"Interval":0}]`
   4. Click **Save** and **Apply**.
3. Adding HVAConfig attribute from OpenText Identity Console.
   1. Go to **Object Management** page and select **NCP server object** to modify.
   2. Under **Valued attribute** click on **"+"** button.
   3. From the list of attributes select HVAConfig.
   4. Enter the value in JSON format and save it.

      Example:`[{"AttributeName":"oidpInstanceData","Type":2,"Limit":1,"Interval":0}]`

HVAConfig can be modified as shown below.

The below sample ldif file depicts modifying the HVAConfig attribute:

```
Version:1
# define attributes
dn: cn=servername, o=server
changetype:modify
replace:HVAConfig
HVAConfig: [{AttributeName":"DirXML-
EntitlementResult","Type":1,"Limit":5010,"Interval":500}],
{"AttributeNAme":"oidpInstaceData","Type":2,"Limit":16588,"Interval":0}]
```

Similarly, we can add and modify multiple attribute in one ldif file.

Sample `ldapmodify` command:

```
ldapmodify -h 10.71.128.217 -D cn=admin,o=novell -w n -f hvaPolicy.ldif
```

In case an attribute mentioned in HVAConfig becomes high valued, the logs regarding the same will be logged under hvAttr-alert.log that will be created in the usual $LOG_DIR. Whenever the configuration is modified, Limber must be triggered from imonitor so that the changes are in effect.

Since HVAConfig is part of NCPServer object, all the attributes monitored are server specific.

Understanding Mandatory and Optional Attributes

Every object has a schema class that has been defined for that type of object, and a class is a group of attributes organized in a meaningful way. Some of these attributes are mandatory and some are optional.

Mandatory Attributes

A mandatory attribute is one that must be filled in when an object is being created. For example, if a new user is being created using the User class, which has the employee number as a mandatory attribute, then the new User object cannot be created without providing the employee number.

Optional Attributes

An optional attribute is one that can be filled in if desired but can be left without content. For example, if a new User object is being created using the User class, which has Other Names as an optional attribute, then the new User object can be created with or without data provided for that attribute, depending on whether the new user is known by other names.

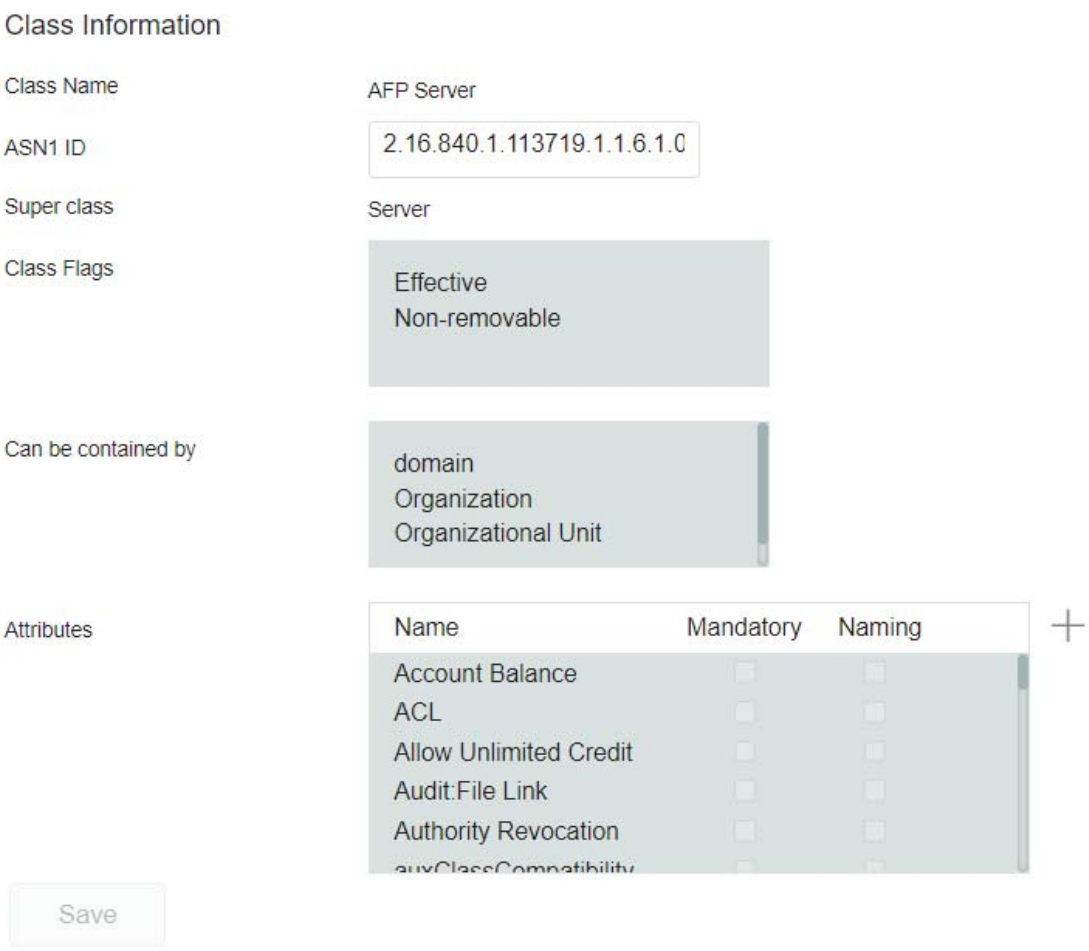
An exception to the rule is when an optional attribute is used for naming, the attribute then becomes mandatory.

Sample Schema

Figure 1-12 is a sample of part of a schema, which might be similar to your base schema. This figure shows information on the Organization class. Most of the information displayed on this screen was specified when the class was created. Some of the optional attributes were added later.

 This icon is assigned to all classes and attributes that are extensions to the base schema.

Figure 1-12 *Class Information Page in OpenText Identity Console*



Designing the Schema

Designing your schema initially can save you time and effort in the long run. You can view the base schema and determine if it will meet your needs or if modifications are required. If changes are needed, use Schema Manager to extend the schema. See "Extending the Schema" on page 132 and "Viewing the Schema" on page 135 for more information.

Partitions

A partition is a logical division of the OpenText eDirectory database. A directory partition forms a distinct unit of data in the tree that stores directory information.

Partitioning allows you to take part of the directory off one server and put it on another server.

If you have slow or unreliable WAN links or your directory has so many objects that the server is overwhelmed and access is slow, you should consider partitioning the directory. For a complete discussion of partitions, see Chapter 6, "Managing Partitions and Replicas," on page 143.

Each directory partition consists of a set of container objects, all the objects contained in them, and data about those objects. OpenText eDirectory partitions don't include any information about the file system or the directories and files contained there.

Partitioning is done with OpenText Identity Console. Partitions are identified in OpenText Identity Console by the following partition icon  in **Partition Management** tile.

Default partitioning for OpenText eDirectory keeps the entire directory together in one partition. When you display the Replica View for a server in Identity Console, any replicas held on that server are shown. For more information, see "Replicas" on page 56 and "Viewing Replicas on an OpenText eDirectory Server" on page 151.

Partitions

Partitions are named by their topmost container. Suppose, there are two partitions, named Tree and Finance. Finance is called a child partition of Tree, because it was split off from Tree. Tree is called the parent partition of Finance.

You might create such a partition because the directory has so many objects that the server is overwhelmed and access to OpenText eDirectory is slow. Creating the new partition allows you to split the database and pass the objects in that branch to a different server.

When you display the Replica View for a partition in OpenText Identity Console, any servers holding a replica of that partition are shown. In this case, Server1 holds a Read-Write replica of the Finance partition. For more information, see "Viewing a Partition's Replicas" on page 153.

Distributing Replicas for Performance

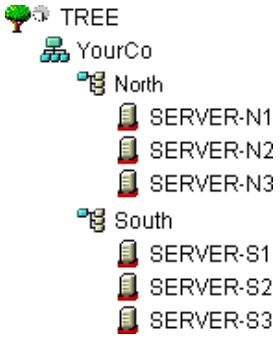
In the preceding example, suppose that Server1 holds replicas of both the Tree partition and the Finance partition. At this point, you haven't gained any performance advantage from OpenText eDirectory because Server1 still holds the entire directory (replicas of both partitions).

To gain the desired performance advantage, you need to move one of the replicas to a different server. For instance, if you move the Tree partition to Server2, then Server2 holds all objects in the Tree and YourCo containers. Server1 holds only objects in the Finance and Accounts containers. The load on both Server1 and Server2 is less than it would be with no partitioning.

Partitions and WAN Links

Suppose your network spans two sites, a North site and a South Site, separated by a WAN link. Three servers are at each site.

Figure 1-13 *Sample Opentext eDirectory Containers*



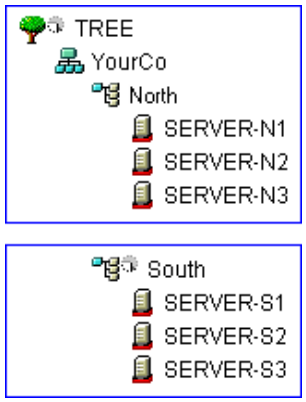
OpenText eDirectory performs faster and more reliably in this scenario if the directory is divided in two partitions.

With a single partition, the replicas are either kept at one site or distributed between the two sites. This proves unwieldy for two reasons:

- If all replicas are kept on servers at the North site, for example, users at the South site encounter delays when logging in or accessing resources. If the link goes down, users at the South site can't log in or access resources at all.
- If replicas are distributed between sites, users can access the directory locally. However, server-to-server synchronization of replicas happens over the WAN link, so there can be OpenText eDirectory errors if the link is unreliable. Any changes to the directory are slow to propagate across the WAN link.

The two-partition solution shown in Figure 1-14 solves performance and reliability problems over the WAN link.

Figure 1-14 *Sample Partitions*



Replicas of the Tree partition are kept on servers at the North site. Replicas of the South partition are kept on servers at the South site, as shown in Figure 1-15.

Figure 1-15 *Sample Partitions, Severs, and Replicas*

Partition	Server	Replica Type
TREE	SERVER-N1	Master
	SERVER-N2	Read/write
	SERVER-N3	Read/write
South	SERVER-S1	Master
	SERVER-S2	Read/write
	SERVER-S3	Read/write

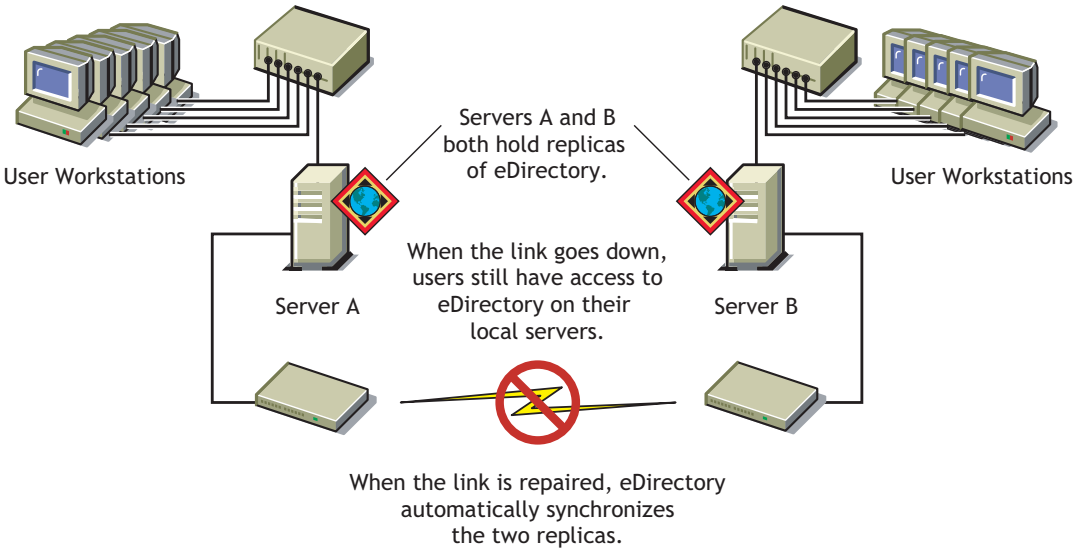
For each site, the objects that represent local resources are kept locally. Synchronization traffic among servers also happens locally over the LAN, rather than over the slow, unreliable WAN link.

OpenText eDirectory traffic is generated over the WAN link, however, when a user or administrator accesses objects at a different site.

Replicas

A replica is a copy or an instance of a user-defined partition that is distributed to an OpenText eDirectory server. If you have more than one OpenText eDirectory server on your network, you can keep multiple replicas (copies) of the directory. That way, if one server or a network link to it fails, users can still log in and use the remaining network resources (see Figure 1-16).

Figure 1-16 *OpenText eDirectory Replicas*



Each server can store more than 65,000 OpenText eDirectory replicas. However, only one replica of the same user-defined partition can exist on the same server. For a complete discussion of replicas, see Chapter 6, “Managing Partitions and Replicas,” on page 143.

We recommend that you keep three replicas for fault tolerance of OpenText eDirectory (assuming you have three OpenText eDirectory servers to store them on). A single server can hold replicas of multiple partitions.

A replica server is a dedicated server that stores only OpenText eDirectory replicas. This type of server is sometimes referred to as a DSMASTER server. This configuration is popular with some companies that use many single-server remote offices. The replica server provides a place for you to store additional replicas for the partition of a remote office location.

It can also be a part of your disaster recovery planning, as described in "Using DSMASTER Servers as Part of Disaster Recovery Planning" on page 430.

OpenText eDirectory replication does not provide fault tolerance for the server file system. Only information about OpenText eDirectory objects is replicated. You can get fault tolerance for file systems by using the Transaction Tracking System™ (TTS™), disk mirroring/duplexing, RAID, or NetIQ Replication Services (NRS).

A master or read/write replica is required on servers that provide bindery services.

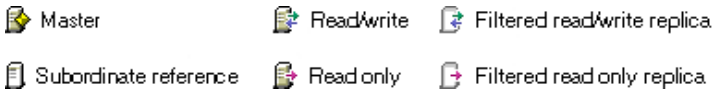
If users regularly access OpenText eDirectory information across a WAN link, you can decrease access time and WAN traffic by placing a replica containing the needed information on a server that users can access locally.

The same is true to a lesser extent on a LAN. Distributing replicas among servers on the network means information is usually retrieved from the nearest available server.

Replica Types

OpenText eDirectory supports the types of replicas shown in the following figure:

Figure 1-17 *Replica Types*



- "Master Replica" on page 57
- "Read/Write Replica" on page 58
- "Read-Only Replica" on page 58
- "Filtered Read/Write Replica" on page 59
- "Filtered Read-Only Replica" on page 59
- "Subordinate Reference Replica" on page 59

Master Replica


The master replica is a writable replica type used to initiate changes to an object or partition. The master replica manages the following types of OpenText eDirectory partition operations:

- Adding replicas to servers
- Removing replicas from servers

- Creating new partitions in the OpenText eDirectory tree
- Removing existing partitions from the OpenText eDirectory tree
- Relocating a partition in the OpenText eDirectory tree

The master replica is also used to perform the following types of OpenText eDirectory object operations:

- Adding new objects to the OpenText eDirectory tree
- Removing, renaming, or relocating existing objects in the OpenText eDirectory tree
- Authenticating objects to the OpenText eDirectory tree
- Adding new object attributes to the OpenText eDirectory tree
- Modifying or removing existing attributes

By default, the first OpenText eDirectory server on your network holds the master replica. There is only one master replica for each partition at a time. If other replicas are created, they are read/write replicas by default.

If you're going to bring down the server holding a master replica for longer than a day or two, you can make one of the read/write replicas the master. The original master replica automatically becomes read/write.

A master replica must be available on the network for OpenText eDirectory to perform operations, such as creating a new replica or creating a new partition.

Read/Write Replica


OpenText eDirectory can access and change object information in a read/write replica as well as the master replica. All changes are then automatically propagated to all replicas.

If OpenText eDirectory responds slowly to users because of delays in the network infrastructure, like slow WAN links or busy routers, you can create a read/write replica closer to the users who need it. You can have as many read/write replicas as you have servers to hold them, although more replicas cause more traffic to keep them synchronized.

Read-Only Replica


The read-only replica is a readable replica type used to read information about all objects in a partition's boundaries. Read-only replicas receive synchronization updates from master and read/write replicas but don't receive changes directly from clients. If login update is enabled then login to read only replica fails as it involves attribute updates.

This replica type is not able to provide bindery emulation, but it does provide OpenText eDirectory tree fault tolerance. If the master replica and all read/write replicas are destroyed or damaged, the read-only replica can be promoted to become the new master replica.

It also provides NDS Object Reads, Fault Tolerance (contains all objects within the Partition boundaries), and NDS Directory Tree Connectivity (contains the Partition Root object).

A read-only replica should never be used to establish a security policy within a tree to restrict the modification of objects, because the client can always access a read/write replica and still make modifications. There are other mechanisms that exist in the directory for this purpose, such as using an Inherited Rights Filter. For more information, see “Inherited Rights Filter (IRF)” on page 68.

Filtered Read/Write Replica

 Filtered read/write replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. The contents are limited to the types of OpenText eDirectory objects and properties specific in the host server's replication filter. Users can read and modify the contents of the replica, and OpenText eDirectory can access and change selected object information. The selected changes are then automatically propagated to all replicas.

With filtered replicas, you can have only one filter per server. This means that any filter defined for a server applies to all filtered replicas on that server. You can, however, have as many filtered replicas as you have servers to hold them, although more replicas cause more traffic to keep them synchronized.

For more information, see “Filtered Replicas” on page 60.

Filtered Read-Only Replica

 Filtered read-only replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. They receive synchronization updates from master and read/write replicas but don’t receive changes directly from clients. Users can read but not modify the contents of the replica. The contents are limited to the types of OpenText eDirectory objects and properties specific in the host server's replication filter.

For more information, see “Filtered Replicas” on page 60.

Subordinate Reference Replica

Subordinate reference replicas are system-generated replicas that don’t contain all the object data of a master or a read/write replica. Subordinate reference replicas, therefore, don’t provide fault tolerance. They are internal pointers that are generated to contain enough information for OpenText eDirectory to resolve object names across partition boundaries.

You can’t delete a subordinate reference replica. OpenText eDirectory deletes it automatically when it is not needed. Subordinate reference replicas are created only on servers that hold a replica of a parent partition but no replicas of its child partitions.

If a replica of the child partition is copied to a server holding the replica of the parent, the subordinate reference replica is automatically deleted.

Filtered Replicas

Filtered replicas contain a filtered set of objects or object classes along with a filtered set of attributes and values for those objects. For example, you might want to create a set of filtered replicas on a single server that contains only User objects from various partitions in the OpenText eDirectory tree. In addition to this, you can choose to include only a subset of the User objects' data (for example, Given Name, Surname, and Telephone Number).

A filtered replica can construct a view of OpenText eDirectory data onto a single server. To do this, filtered replicas let you create a scope and a filter. This results in an OpenText eDirectory server that can house a well-defined data set from many partitions in the tree.

The descriptions of the server's scope and data filters are stored in OpenText eDirectory and can be managed through the Server object in OpenText Identity Console.

A server hosting one of more filtered replicas has only a single replication filter. Therefore, all filtered replicas on the server contain the same subset of information from their respective partitions. The master partition replica of a filtered replica must be hosted on an OpenText eDirectory server running OpenText eDirectory 8.5 or later.

Filtered replicas can

- Reduce synchronization traffic to the server by reducing the amount of data that must be replicated from other servers.
- Reduce the number of events that must be filtered by OpenText Identity Manager.

  For more information on OpenText Identity Manager, see the OpenText™ Identity Manager CE 24.4 (v4.10) Overview and Planning Guide.
- Reduce the size of the directory database.

  Each replica adds to the size of the database. By creating a filtered replica that contains only specific classes (instead of creating a full replica), you can reduce the size of your local database.

  For example, if your tree contains 10,000 objects but only a small percentage of those objects are Users, you could create a filtered replica containing only the User objects instead of a full replica containing all 10,000 objects.

Other than the ability to filter data stored in a local database, the filtered replica is like a normal OpenText eDirectory replica and it can be changed back to a full replica at any time.

NOTE: Filtered replicas by default will have the Organization and the Organizational Unit as mandatory filters.

For more information on setting up and managing filtered replicas, see "Setting Up and Managing Filtered Replicas" on page 150.

Allowing Local Logins to Filtered Replicas

In addition to selecting the **Enable local login** option in OpenText Identity Console, to allow local logins to a Filtered Replica, you should also add the class ndsLoginProperties to the filter.

Before logging into the filtered replica, you must set the following attributes:

- Detect Intruder

- Intruder Attempt Reset Interval
- Last Login Time
- Locked By Intruder
- Lockout After Detection
- Login Allowed Time Map
- Login Disabled
- Login Expiration Time
- Login Grace Limit
- Login Grace Remaining
- Login Intruder Address
- Login Intruder Attempts
- Login Intruder Limit
- Login Intruder Reset Time
- Login Maximum Simultaneous
- Login Time
- Network Address
- Network Address Restriction
- Password Expiration Interval
- Password Expiration Time
- Private Key
- Public Key
- nspmDoNotExpirePassword
- nspmPasswordKey
- nspmPasswordPolicyDN
- pwdAccountLockedTime
- pwdFailureTime
- sasLoginFailureDelay
- sasOTPCounter
- sasOTPDigits
- sasOTPEnabled
- sasOTPReSync
- sasUpdateLoginInfo
- sasUpdateLoginTimeInterval

NOTE: The above attributes can be set on the user object, parent container or login policy.

Server Synchronization in the Replica Ring

When multiple servers hold replicas of the same partition, those servers are considered a replica ring. Synchronization is the propagation of directory information from one replica to another, so the information in each partition is consistent with the other. OpenText eDirectory automatically keeps those servers synchronized. For more information, refer "Synchronization" on page 113

The following are the types of OpenText eDirectory synchronization:

- Normal Synchronization or Replica Synchronization
- Priority Sync

Access to Resources

OpenText eDirectory provides a basic level of network access security through default rights. You can provide additional access control by completing the tasks outlined below.

- Assigning rights

  Each time a user attempts to access a network resource, the system calculates the user's effective rights to that resource. To ensure that users have the appropriate effective rights to resources, you can make explicit trustee assignments, grant security equivalences, and filter inherited rights.

  To simplify the assignment of rights, you can create Group and Organizational Role objects, then assign users to the groups and roles.

- Adding login security

  Login security is not provided by default. You can set up several optional login security measures, including login passwords, login location and time restrictions, limits on concurrent login sessions, intruder detection, and login disabling.

- Setting up role-based administration

  You can set up administrators for specific object properties and grant them rights to only those properties. This allows you to create administrators with specific responsibilities that can be inheritable to subordinates of any given container object. A role-based administrator can have responsibilities over any specific properties, such as those that relate to employee information or passwords.

  For instructions on setting up RAC Configuration, see configuring RAC in the *OpenText™ Identity Console Administration Guide* (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

  You can also define roles in terms of the specific tasks that administrators can perform in role-based administration applications. See Configuring Roles and Access Control for more information.

OpenText eDirectory Rights

When you create a tree, the default rights assignments give your network generalized access and security. Some of the default assignments are as follows:

- User Admin has the Supervisor right to the top of the tree, giving Admin complete control over the entire directory. Admin also has the Supervisor right to the Server object, giving complete control over any volumes on that server.
- [Public] has the Browse right to the top of the tree, giving all users the right to view any objects in the tree.
- Objects created through an upgrade process, printing upgrade, or Windows user migration receive trustee assignments appropriate for most situations.

Trustee Assignments and Targets

The assignment of rights involves a trustee and a target object. The trustee represents the user or set of users that are receiving the authority. The target represents those network resources the users have authority over.

- If you make an Alias a trustee, the rights apply only to the object the alias represents. The Alias object can be an explicit target, however.
- A file or directory in the file system can also be a target, although file system rights are stored in the file system itself, not in OpenText eDirectory.

NOTE: The [Public] trustee is not an object. It is a specialized trustee that represents any network user, logged in or not, for rights assignment purposes.

[This] is a special type of trustee, that is defined to be an authenticated object, when its name matches the entry being accessed. This helps the administrator to easily specify rights such as, every user manages his own telephone number, with a single ACL at the top of the tree with [This] as a trustee.

OpenText eDirectory Rights Concepts

The following concepts can help you better understand OpenText eDirectory rights.

- “Object (Entry) Rights” on page 64
- “Property Rights” on page 64
- “Effective Rights” on page 65
- “How Effective Rights Are Calculated” on page 65
- “Security Equivalence” on page 67
- “Access Control List (ACL)” on page 68
- “Inherited Rights Filter (IRF)” on page 68

Object (Entry) Rights

When you make a trustee assignment, you can grant object rights and property rights. Object rights apply to manipulation of the entire object, while property rights apply only to certain object properties. An object right is described as an entry right because it provides an entry into the OpenText eDirectory database.

A description of each object right follows:

- **Supervisor** includes all rights to the object and all of its properties.
- **Browse** lets the trustee see the object in the tree. It does not include the right to see an object's properties.
- **Create** applies only when the target object is a container. It allows the trustee to create new objects below the container and also includes the Browse right.
- **Delete** lets the trustee delete the target from the directory.
- **Rename** lets the trustee change the name of the target.

Property Rights

When you make a trustee assignment, you can grant object rights and property rights. Object rights apply to manipulation of the entire object, while property rights apply only to certain object properties.

OpenText Identity Console gives you two options for managing property rights:

- You can manage all properties at once when the **[All Attributes Rights]** item is selected.
- You can manage one or more individual properties when the specific property is selected.

IMPORTANT: If you grant a trustee Read access to the [All Attributes Rights] property of a user, the trustee is granted Read access to the `Password Management` attribute for that user. The trustee can then read the user information such as password-related information, password-expiration date, grace login limit and so on.

For more information about creating and managing password policies, see "Creating Password Policies" on page 722.

A description of each property right follows:

- **Supervisor** gives the trustee complete power over the property.
- **Compare** lets the trustee compare the value of a property to a given value. This right allows searching and returns only a true or false result. It does not allow the trustee to actually see the value of the property.
- **Read** lets the trustee see the values of a property. It includes the Compare right.
- **Write** lets the trustee create, change, and delete the values of a property.
- **Add Self** lets the trustee add or remove itself as a property value. It only applies to properties with object names as values, such as membership lists or Access Control Lists (ACLs).

Effective Rights

Users can receive rights in a number of ways, such as explicit trustee assignments, inheritance, and security equivalence. Rights can also be limited by Inherited Rights Filters and changed or revoked by lower trustee assignments. The net result of all these actions—the rights a user can employ—are called *effective rights*.

A user's effective rights to an object are calculated each time the user attempts an action.

How Effective Rights Are Calculated

Each time a user attempts to access a network resource, OpenText eDirectory calculates the user's effective rights to the target resource using the following process:

1. OpenText eDirectory lists the trustees whose rights are to be considered in the calculation. These include
   - The user who is attempting to access the target resource.
   - The objects that the user is security equivalent to.
2. For each trustee in the list, OpenText eDirectory determines its effective rights as follows:

   2a OpenText eDirectory starts with the inheritable rights that the trustee has at the top of the tree.

   OpenText eDirectory checks the Object Trustees (ACL) property of the Tree object for entries that list the trustee. If any are found and they are inheritable, OpenText eDirectory uses the rights specified in those entries as the initial set of effective rights for the trustee.

   2b OpenText eDirectory moves down a level in the branch of the tree that contains the target resource.

   2c OpenText eDirectory removes any rights that are filtered at this level.

   OpenText eDirectory checks the ACL at this level for Inherited Rights Filters (IRFs) that match with the right types (object, all properties, or a specific property) of the trustee's effective rights. If any are found, OpenText eDirectory removes from the trustee's effective rights any rights that are blocked by those IRFs.

   For example, if the trustee's effective rights so far include an assignment of Write All Properties, but an IRF at this level blocks Write All Properties, the system removes Write All Properties from the trustee's effective rights.

   2d OpenText eDirectory adds any inheritable rights that are assigned at this level, overriding as needed.

   OpenText eDirectory checks the ACL at this level for entries that list the trustee. If any are found, and they are inheritable, it copies the rights from those entries to the trustee's effective rights, overriding as needed.

   For example, if the trustee's effective rights so far include the Create and Delete object rights but no property rights, and if the ACL at this level contains both an assignment of zero object rights and an assignment of Write all properties for this trustee, then the system replaces the trustee's existing object rights (Create and Delete) with zero rights and adds the new all property rights.

   2e OpenText eDirectory repeats the filtering and adding steps (Step 2c and Step 2d above) at each level of the tree, including at the target resource.

2f OpenText eDirectory adds any non-inheritable rights assigned at the target resource, overriding as needed.

OpenText eDirectory uses the same process as in Step 2d above. The resulting set of rights constitutes the effective rights for this trustee.

3 OpenText eDirectory combines the effective rights of all the trustees in the list as follows:

3a OpenText eDirectory includes every right held by any trustee in the list and excludes only those rights that are missing from every trustee in the list. OpenText eDirectory does not mix right types. For example, it does not add rights for a specific property to rights for all properties or vice versa.

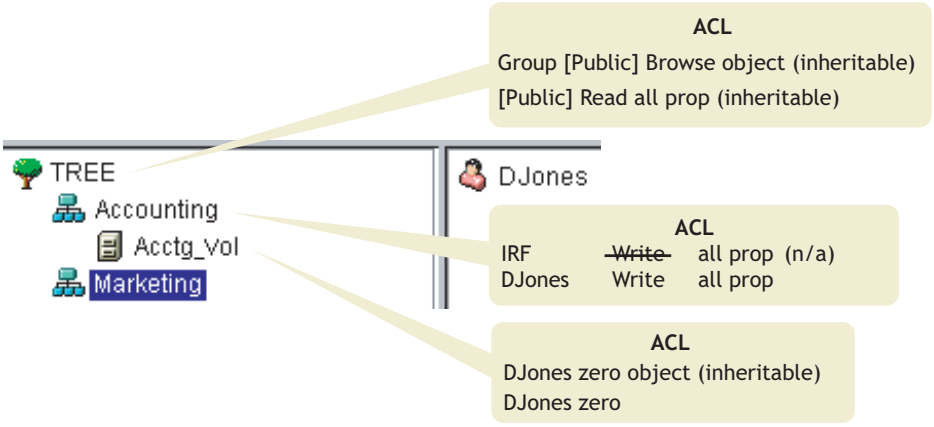
3b OpenText eDirectory adds rights that are implied by any of the current effective rights.

The resulting set of rights constitutes the user's effective rights to the target resource.

Example

User DJones is attempting to access volume Acctg_Vol. See Figure 1-18.

Figure 1-18 *Sample Trustee Rights*



The following process shows how OpenText eDirectory calculates DJones' effective rights to Acctg_Vol:

1. The trustees whose rights are to be considered in the calculation are DJones, Marketing, Tree, and [Public].

   This assumes that DJones doesn't belong to any groups or roles and has not been explicitly assigned any security equivalences.

2. The effective rights for each trustee are as follows:

   - DJones: Zero object, zero all properties

     The assignment of zero all property rights at Acctg Vol overrides the assignment of Write all properties at Accounting.

   - Marketing: Zero all properties

     The assignment of Write all properties at the top of the tree is filtered out by the IRF at Accounting.

   - Tree: No rights

     No rights are assigned for Tree anywhere in the pertinent branch of the tree.

- [Public]: Browse object, Read all properties

  These rights are assigned at the root and aren't filtered or overridden anywhere in the pertinent branch of the tree.

3. Combining the rights from all these trustees results in the following:

   DJones: Browse object, Read all properties

4. Adding the Compare all properties right that is implied by the Read all properties right, DJones has the following final effective rights to Acctg_Vol:

   DJones: Browse object, Read and Compare all properties

Blocking Effective Rights

Because of the way that effective rights are calculated, it is not always obvious how to block particular rights from being effective for specific users without resorting to an IRF (an IRF blocks rights for all users).

To block particular rights from being effective for a user without using an IRF, do either of the following:

- Ensure that neither the user nor any of the objects that the user is security equivalent to ever gets assigned those rights, either at the target resource or at any level above the target resource in the tree.
- If the user or any object that the user is security equivalent to does get assigned those rights, ensure that that object also has an assignment lower in the tree that omits those rights. Do this for every trustee (associated with the user) that has the unwanted rights.

Security Equivalence

Security equivalence means having the same rights as another object. When you make one object security equivalent to another object, the rights of the second object are added to the rights of the first object when the system calculates the first object's effective rights.

For example, suppose you make User object Joe security equivalent to the Admin object. After you create the security equivalence, Joe has the same rights to the tree and file system as Admin.

There are three types of security equivalence:

- Explicit: By assignment
- Automatic: By membership in a group or role
- Implied: Equivalent to all parent containers and the [Public] trustee

Security equivalence is effective only for one step. For example, if you make a third user security equivalent to Joe in the example above, that user does not receive Admin rights.

Security equivalence is recorded in OpenText eDirectory as values in the User object's Security Equal To property.

When you add a User object as an occupant to an Organizational Role object, that User automatically becomes security equivalent to the Organizational Role object. The same is true when a User becomes a member of a Group role object.

Access Control List (ACL)

The Access Control List (ACL) is also called the Object Trustees property. Whenever you make a trustee assignment, the trustee is added as a value to the Object Trustees (ACL) property of the target.

This property has strong implications for network security for the following reasons:

- Anyone who has the Supervisor or Write right to the Object Trustees (ACL) property of an object can determine who is a trustee of that object.
- Any users with the Add Self right to the Object Trustees (ACL) property of an object can change their own rights to that object. For example, they can grant themselves the Supervisor right.

For these reasons, be careful giving Add Self rights to all properties of a container object. That assignment makes it possible for the trustee to become Supervisor of that container, all objects in it, and all objects in containers beneath it.

Inherited Rights Filter (IRF)

The Inherited Rights Filter allows you to block rights from flowing down the OpenText eDirectory Tree. For more information on configuring this filter, see "Blocking Inherited Rights to an OpenText eDirectory Object or Property" on page 73.

Default Rights for a New Server

When you install a new Server object into a tree, the following trustee assignments are made:

Default Trustees	Default Rights
Admin (first OpenText eDirectory server in the tree)	Supervisor object right to the Tree object. Admin has the Supervisor object right to the Server object, which means that Admin also has the Supervisor right to the root directory of the file system of any volumes on the server.
[Public] (first OpenText eDirectory server in the tree)	Browse object right to the Tree object.
Tree	The Tree Read property right to the Host Server Name and Host Resource properties on all Volume objects. This gives all objects access to the physical volume name and physical server name.
Container objects	Read and File Scan rights to the `sys:\public` folder. This allows User objects under the container to access utilities in `\public`. **NOTE:** These rights only apply to servers running OES Linux.
User objects	If home directories are automatically created for users, the users have the Supervisor right to those directories.

Delegated Administration

OpenText eDirectory lets you delegate administration of a branch of the tree, revoking your own management rights to that branch. One reason for this approach is that special security requirements require a different administrator with complete control over that branch.

To delegate administration:

1 Grant the Supervisor object right to a container.

1a On the OpenText Identity Console home page, click **Rights Management** tile.

1b Click **Modify Trustees**.

1c Click Search Object  and search context of the container object that you want to control access to, then click **OK**.

1d Click **Assigned Rights**.

1e Click the **Supervisor** check box for the properties you want.

1f Click **Done**, then click **Apply**.

2 Create an IRF on the container that filters the Supervisor and any other rights you want blocked.

2a On the OpenText Identity Console home page, click **Rights Management** tile.

2b Click **Inherited Rights Filter**.

2c Specify the name and context of the object whose inherited rights filter you want to modify, then click **OK**.

2d Edit the list of inherited rights filters as needed.

To edit the list of filters, you must have the Supervisor or Access Control right to the ACL property of the object. You can set filters that block inherited rights to the object as a whole, to all the properties of the object, and to individual properties.

NOTE: These filters won't block rights that are explicitly granted a trustee on this object, since such rights aren't inherited.

2e Click **OK**.

IMPORTANT: If you delegate administration to a User object and that object is subsequently deleted, there are no objects with rights to manage that branch.

To delegate administration of specific OpenText eDirectory properties, such as Password Management, see “Granting Equivalence” on page 71.

To delegate the use of specific functions in role-based administration applications, see “Configuring Roles and Access Control” on page 106.

Administering Rights

- “Assigning Rights Explicitly” on page 70
- “Granting Equivalence” on page 71
- “Blocking Inherited Rights to an OpenText eDirectory Object or Property” on page 73
- “Viewing Effective Rights to an OpenText eDirectory Object or Property” on page 73

Assigning Rights Explicitly

When the default rights assignments in your OpenText eDirectory tree provide users with either too much or not enough access to resources, you can create or modify explicit rights assignments. When you create or modify a rights assignment, you start by selecting either the resource that you are controlling access to or the trustee (the OpenText eDirectory object that possesses, or will possess, the rights).

TIP: To manage users' rights collectively rather than individually, make a group, role, or container object the trustee. To restrict access to a resource globally (for all users), see "Blocking Inherited Rights to an OpenText eDirectory Object or Property" on page 73.

- "Controlling Access to OpenText eDirectory by Resource" on page 70
- "Controlling Access to OpenText eDirectory by Trustee" on page 70

Controlling Access to OpenText eDirectory by Resource

1. On the OpenText Identity Console home page, click **Rights Management** tile.
2. Click **Modify Trustees**.
3. Specify the name and context of the OpenText eDirectory resource (object) that you want to control access to, then click **OK**.

   Choose a container if you want to control access to all the objects below it.
4. Edit the list of trustees and their rights assignments as needed.

   4a To modify a trustee's rights assignment, select the trustee, click **Assigned Rights**, modify the rights assignment as needed, then click **Done**.

   4b To add an object as a trustee, click **Add Trustee**, select the object, click **OK**, click **Assigned Rights** to assign the trustee's rights, then click **Done**.

   When creating or modifying a rights assignment, you can grant or deny access to the object as a whole, to all the properties of the object, and to individual properties.

   4c To remove an object as a trustee, select the trustee, then click **Delete Trustee**.

   The deleted trustee no longer has explicit rights to the object or its properties but might still have effective rights through inheritance or security equivalence.
5. Click **OK**.

Controlling Access to OpenText eDirectory by Trustee

1. On the Identity Console home page, click **Rights Management** tile > **Trustee**.
2. Click **Rights** > **Objects Name** for selecting the context.
3. Enter the name and context of the trustee (the object that possesses, or will possess, the rights) whose rights you want to modify.
4. In the **Context to Search From** field, specify the part of the OpenText eDirectory tree to be searched for OpenText eDirectory objects that the trustee currently has rights assignments to.
5. Click **OK**.

   A screen appears with the results of the search.

6 Edit the trustee's OpenText eDirectory rights assignments as needed.

6a To add a rights assignment, click **Assigned Rights**, assign the trustee's rights, then click **Done**.

6b To modify a rights assignment, select the object you want to control access to, click **Assigned Rights**, modify the trustee's rights assignment as needed, then click **Done**.

When creating or modifying a rights assignment, you can grant or deny access to the object as a whole, to all the properties of the object, and to individual properties.

6c To remove a rights assignment, select the object you want to control access to, then click **Delete Properties**.

The trustee no longer has explicit rights to the object or its properties but might still have effective rights through inheritance or security equivalence.

7 Click **Apply**.

Granting Equivalence

A user who is security equivalent to another OpenText eDirectory object effectively has all the rights of that object. A user is automatically security equivalent to the groups and roles that they belong to. All users are implicitly security equivalent to the [Public] trustee and to each container above their User objects in the OpenText eDirectory tree, including the Tree object. You can also explicitly grant a user security equivalence to any OpenText eDirectory object.

NOTE: The tasks in this section allow you to delegate administrative authority through OpenText eDirectory rights. If you have administration applications that use Role-Based Services (RBS) roles, you can also delegate administrative authority by assigning users membership in those roles.

- “Granting Security Equivalence by Membership” on page 71
- “Granting Security Equivalence Explicitly” on page 72
- “Setting Up an Administrator For an Object's Specific OpenText eDirectory Properties” on page 72

Granting Security Equivalence by Membership

1 If you haven't already done so, create the group or role object that you want the users to be security equivalent to.

See “Creating an Object” on page 97 for details.

2 Grant the group or role the OpenText eDirectory rights that you want the users to have.

See “Assigning Rights Explicitly” on page 70 for details.

3 Edit the membership of the group or role to include those users who need the rights of the group or role.

- For a Group object, use the **Modify Groups** window.

  On the OpenText Identity Console home page, click **Groups Management** tile > Select the Object. Click **Modify Object** icon to specify the members you want to add to the group and click **Save**.

- For a Role object, use the **Modify Object** window.

  On the OpenText Identity Console home page, click **Object Management** > On the **Type** drop down menu select `Organizational Role` object, and click **Search**. Select the Object, click **Valued Attribute**, select any attributes and click **OK**. On the **Modify Object** window, specify the selected attribute details you want to add to the role and click **Save**.

4 Click **OK**.

Granting Security Equivalence Explicitly

1 On the OpenText Identity Console home page, click **Object Management** > On the **Type** drop down menu select `User` > click **Search**.

2 Click on searched user and click on **Security** tab.

3 Under **Security** tab, grant the security equivalence as follows:

- If you chose a user, click **Security Equal To**, select or browse to the name and context of the object that you want the user to be equivalent in terms of security, then click **OK**.
- If you chose an object that you want the user to be security equivalent to, click **Security Equal To Me**, select or browse to the name and context of the user that you want the object to be equivalent to in terms of security, then click **OK**.

The contents of these two property pages are synchronized by the system.

4 Click **Save**.

5 Click **OK**.

Setting Up an Administrator For an Object's Specific OpenText eDirectory Properties

1 If you haven't already done so, create the User, Group, Role, or Container object that you want to make a trustee of the object's specific properties.

If you create a container as a trustee, all objects inside and below the container will have the rights you grant. You must make the property inheritable or the container and its members will not have rights below its level.

See "Creating an Object" on page 97 for information.

2 On the OpenText Identity Console home page, click **Rights Management** tile.

3 Click **Modify Trustees**.

4 Click **Search** and select the name and context of the highest-level container that you want the administrator to manage, then click **OK**.

5 On the Modify Trustees page, click **Add Trustee**, select the object that represents the administrator, then click **OK**.

6 Click **Assigned Rights** for the trustee you just added, then click **Add Property**.

7 Select the properties you want to add to the property list, then click **OK**.

8 For each property that the administrator will manage, assign the needed rights.

Be sure to select the **Inheritable** check box on each rights assignment.

9 Click **Done**, then click **Apply**.

Blocking Inherited Rights to an OpenText eDirectory Object or Property

In OpenText eDirectory, rights assignments on containers can be inheritable or non-inheritable. In the file system, all rights assignments on folders are inheritable. In OpenText eDirectory, you can block such inheritance on individual subordinate items so that the rights aren't effective on those items, no matter who the trustee is.

1. On the OpenText Identity Console home page, click **Rights Management** tile.
2. Click **Modify Inherited Rights Filter**.
3. Click **Search**  and select the context of the object whose inherited rights filter you want to modify, then click **OK**.

   This displays a list of the inherited rights filters that have already been set on the object.
4. On the property page, edit the list of inherited rights filters as needed.

   To edit the list of filters, you must have the Supervisor or Access Control right to the ACL property of the object. You can set filters that block inherited rights to the object as a whole, to all the properties of the object, and to individual properties.

   NOTE: These filters won't block rights that are explicitly granted a trustee on this object, because such rights aren't inherited.

   ---
5. Click **Apply**.

Viewing Effective Rights to an OpenText eDirectory Object or Property

Effective rights are the actual rights users can exercise on specific network resources. They are calculated by OpenText eDirectory based on explicit rights assignments, inheritance, and security equivalence. You can query the system to determine a user's effective rights to any resource.

1. On the OpenText Identity Console home page, click **Rights Management** tile.
2. Click **Effective Rights**.
3. Click **Search**  and select the context of the object whose inherited rights filter you want to modify, then click **OK**.
4. Choose from the following options:

Option	Description
Property Name	Lists the properties that the trustee has effective rights to. The properties are read from OpenText eDirectory and so are always shown in English. Each item in the list is one of the following types: [All Attributes Rights]-Represents all the properties of the object. [Entry Rights]-Represents the object as a whole. Rights to this item don't imply any property rights, except in the case of Supervisor. Specific properties-These are specific properties that the trustee has rights to individually. By default, only properties of this object class are listed (see below).
Effective Rights	Shows the trustee's effective rights to the selected property, as calculated by OpenText eDirectory.
Show All Properties in Schema	Leave this check box deselected to show only the properties of this object class. To show the properties of all classes defined in the OpenText eDirectory schema, select this check box. The additional properties are pertinent only if this object is a container, or if it has been extended to include the properties of an auxiliary class. The additional properties are shown without a bullet next to them.

5 Click **Done**.

2 Designing Your OpenText eDirectory Network

The design of OpenText eDirectory impacts virtually every network user and resource. A good OpenText eDirectory design can enhance the performance and value of the entire network by making the network more efficient, fault tolerant, secure, and scalable, and operable. This chapter provides suggestions for designing your OpenText eDirectory network.

- "OpenText eDirectory Design Basics" on page 75
- "Designing the OpenText eDirectory Tree" on page 76
- "Guidelines for Partitioning Your Tree" on page 82
- "Guidelines for Replicating Your Tree" on page 84
- "Planning the User Environment" on page 86
- "Designing OpenText eDirectory for e-Business" on page 87
- "Understanding the OpenText Certificate Server" on page 88
- "Synchronizing Network Time" on page 92

OpenText eDirectory Design Basics

An efficient OpenText eDirectory design is based on the network layout, organizational structure of the company, and proper preparation.

If you are designing OpenText eDirectory for e-business, refer to "Designing OpenText eDirectory for e-Business" on page 87.

Network Layout

The network layout is the physical setup of your network. To develop an efficient OpenText eDirectory design, you need to be aware of the following:

- WAN links
- Users that need remote access
- Network resources (such as number of servers)
- Network conditions (such as frequent power outages)
- Anticipated changes to the network layout

Organizational Structure

The organizational structure of the company will influence the OpenText eDirectory design. To develop an efficient OpenText eDirectory design you need,

- The organizational chart and an understanding of how the company operates.
- Personnel who have the skills needed to complete the design and implementation of your OpenText eDirectory tree.

  You will need to identify personnel who can do the following:

  - Maintain the focus and schedule of the OpenText eDirectory design
  - Understand OpenText eDirectory design, design standards, and security
  - Understand and maintain the physical network structure
  - Manage the internetwork backbone, telecommunications, WAN design, and router placement

Preparing for OpenText eDirectory Design

Before you actually create the OpenText eDirectory design, you should

- Set realistic expectations concerning scope and schedule.
- Notify all users who will be affected by the design of your implementation of OpenText eDirectory.
- Review the information in "Network Layout" on page 75 and "Organizational Structure" on page 76.

Designing the OpenText eDirectory Tree

Designing the OpenText eDirectory tree is the most important procedure in the design and implementation of a network. The design consists of the following tasks:

- "Creating a Naming Standards Document" on page 76
- "Designing the Upper Layers of the Tree" on page 79
- "Designing the Lower Layers of the Tree" on page 81

Creating a Naming Standards Document

Using standard names such as object names makes your network more intuitive to both users and administrators. Written standards can also specify how administrators set other property values, such as telephone numbers and addresses.

Searching and browsing the directory rely greatly on the consistency of naming or property values.

The use of standard names also makes it easier for OpenText Identity Manager to move data between OpenText eDirectory and other applications. For more information about OpenText Identity Manager, see the *OpenText Identity Manager* (https://www.netiq.com/documentation/identity-manager-4.10/).

Naming Conventions

- "Objects" on page 77
- "Server Objects" on page 77
- "Country Objects" on page 77

Objects

- The name must be unique in the container. For example, Debra Jones and Daniel Jones cannot both be named DJONES if they are in the same container.
- Special characters are allowed. However, plus signs (+), equals signs (=), and periods (.) must be preceded by a backslash (\) if used. Additional naming conventions apply to Server and o, as well as to bindery services and multilingual environments.
- Uppercase and lowercase letters, as well as underscores and spaces, are displayed as you first entered them, but they aren't distinguished. For example, `Manager_Profile` and `MANAGER PROFILE` are considered to be identical.
- If you use spaces, you must enclose the name in quotes when entering it on the command line or in login scripts.

Server Objects

- Server objects are automatically created when you install new servers.
- You can create additional Server objects for existing Windows servers and for OpenText eDirectory servers in other trees, but they are all treated as bindery objects.
- When creating a Server object, the name must match the physical server name, which
  - Is unique in the entire network.
  - Is from 2 to 47 characters long.
  - Contains only letters A-Z, numbers 0-9, hyphens (-), periods (.), and underscores (_).
  - Does not use a period as the first character.
- Once named, the Server object cannot be renamed in OpenText Identity Console. If you rename it at the server, the new name automatically appears in OpenText Identity Console.

Country Objects

Country objects should follow the standard two-letter ISO country code.

For more information, see the ISO 3166 Code Lists (https://www.iso.org/iso-3166-country-codes.html).

Multilingual Considerations

If you have workstations running in different languages, you might want to limit object names to characters that are viewable on all the workstations. For example, a name entered in Japanese cannot contain characters that aren't viewable in Western languages.

IMPORTANT: The Tree name should always be specified in English.

Sample Standards Document

The following is a sample document containing standards for some of the most frequently used properties. You need to have standards only for those properties you use. Distribute the standards document to all administrators responsible for creating or modifying objects.

Object Class | Property	Standard	Examples	Rationale
User | Login name	First initial, middle initial (if applicable), and last name (all lowercase). Eight characters maximum. All common names are unique in the company.	msmith, bjohnson	Using unique names company-wide is not required by OpenText eDirectory but helps avoid conflicts within the same context (or bindery context).
User | Last name	Last name (normal capitalization).	Smith	Used for generating mailing labels.
Telephone and fax numbers	Numbers separated by hyphens.	US: 123-456-7890 Other: 44-344-123456	Used by autodialing software.
Multiple classes | Location	Two-letter location code (uppercase), hyphen, mail stop.	BA-C23	Used by interoffice mail carriers.
Organization | Name	The name of your company for all trees.	YourCo	If you have separate trees, a standard Organization name allows for future merging of trees.
Organizational Unit | Name (based on location)	Two- or three-letter location code, all uppercase.	ATL, CHI, CUP, LA, BAT, BOS, DAL	Short, standard names are used for efficient searching.
Organizational Unit | Name (based on department)	Department name or abbreviation.	Sales, Eng	Short, standard names make it easy to identify which department the container is servicing.
Group | Name	Descriptive name.	Project Managers	Avoid extremely long names. Some utilities will not display them.
Directory Map | Name	Contents of the directory indicated by the Directory Map.	DOSAPPS	Short, standard names make it easy to identify which department the container is servicing.
Profile | Name	Purpose of the profile.	MobileUser	Short, standard names make it easy to identify which department the container is servicing.

Object Class | Property	Standard	Examples	Rationale
Server | Name	SERV, hyphen, department, hyphen, unique number.	SERV-Eng-1	OpenText eDirectory requires server names to be unique in the tree.

Designing the Upper Layers of the Tree

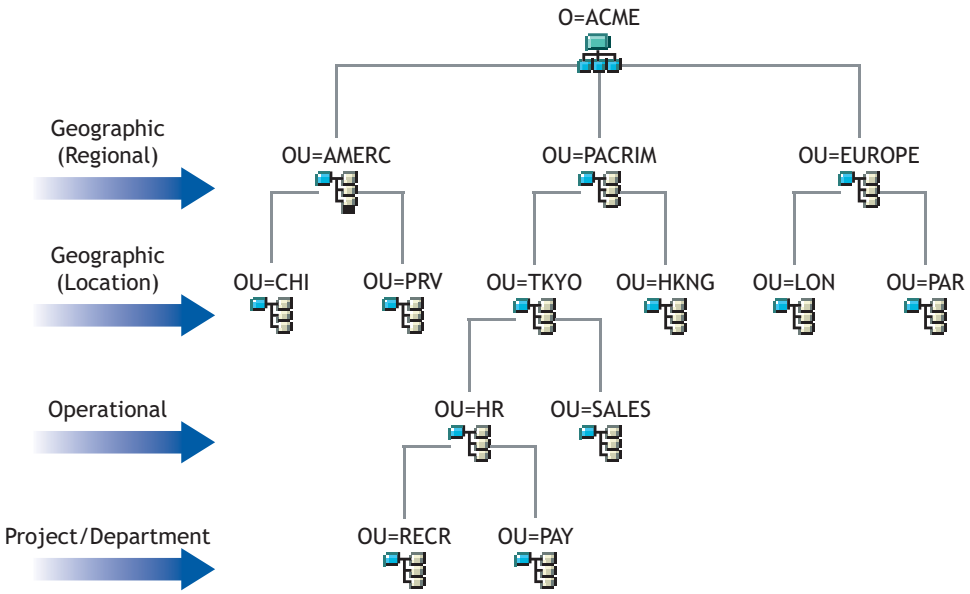
You should carefully design the upper layers of the tree because changes to the upper layers affect the rest of the tree, especially if your organization has WAN links. You want to design the top of the tree so that few changes will be necessary.

Use the following OpenText eDirectory design rules to create your OpenText eDirectory tree:

- Use a pyramid design.
- Use one OpenText eDirectory tree with a unique name.
- Create a single Organization object.
- Create first-level Organizational Units that represent the physical network infrastructure.

Figure 2-1 depicts the OpenText eDirectory design rules.

Figure 2-1 *OpenText eDirectory Design Rules*



To create the upper layers of the tree, see “Creating an Object” on page 97 and “Modifying an Object's Properties” on page 98.

Using a Pyramid Design

With a pyramid-designed OpenText eDirectory, managing, initiating changes to large groups, and creating logical partitions are easier.

The alternative to the pyramid design is a flat tree that places all objects in the top layers of the tree. OpenText eDirectory can support a flat tree design. However, a flat tree design can be more difficult to manage and partition.

Using One OpenText eDirectory Tree with a Unique Name

A single tree works best for most organizations. By default, one tree is created. With one tree you have single-user identity on the network, simpler administration of security, and single point of management.

This recommendation for a single tree for business use does not preclude additional trees for testing and development.

Some organizations, however, might need multiple trees because of legal, political, or corporate issues. For example, an organization consisting of several autonomous organizations might need to create several trees. If your organization needs multiple trees, consider using OpenText Identity Manager to simplify management. For more information about OpenText Identity Manager, see the (https://www.netiq.com/documentation/identity-manager-4.10/).

When you name the tree, use a unique name that will not conflict with other tree names. Use a name that is short and descriptive, such as EDL-TREE.

If two trees have the same name and are located on the same network, you might encounter the following problems:

- Updates going to the wrong tree
- Resources disappearing
- Rights disappearing
- Corruption

You can change the tree name using the DSMerge utility, but do so with caution. A tree name change impacts the network because you need to reconfigure the clients to use the new tree name.

Creating a Single Organization Object

Generally, an OpenText eDirectory tree should have one Organization object. By default, a single Organization object is created and named after your company. This allows you to configure changes that apply to the whole company from a single location in the tree.

For example, you can use ZENworks® to create a Workstation Import Policy object in the Organization object. In this policy, which affects the whole organization, you define how Workstation objects are named when created in OpenText eDirectory.

In the Organization container, the following objects are created:

- Admin
- Server

- Volume

  Networks with only a Windows or Linux server running OpenText eDirectory have no Volume objects.

You might want to create multiple Organization objects if your company has the following needs:

- It comprises multiple companies that do not share the same network.
- It needs to represent separate business units or organizations.
- It has a policy or other internal guidelines that dictate that organizations remain separate.

Creating Organizational Units That Represent the Physical Network

First-level Organizational Unit design is important because it affects the partitioning and efficiency of OpenText eDirectory.

For networks that span more than one building or location using either a LAN or a WAN, the first-level Organizational Unit object design should be based on location. This allows you to partition OpenText eDirectory in a way that keeps all objects in a partition at one location. It also provides a natural place to make security and administrator assignments for each location.

Designing the Lower Layers of the Tree

You should design the lower layers of the tree based on the organization of network resources. You have more freedom in designing the lower layers of an OpenText eDirectory tree than the upper layers because lower-layer design affects only objects at the same location.

To create the lower layers of the tree, see “Creating an Object” on page 97 and “Modifying an Object's Properties” on page 98.

Determining Container, Tree, and Database Size

The number of lower-level container objects you create depends on the total number of objects in your tree and your disk space and disk I/O speed limitations. eDirectory has been tested with over 1 billion objects in a single OpenText eDirectory tree, so the only real limitations are disk space, disk I/O speed, and RAM to maintain performance. Keep in mind that the impact of replication on a large tree is significant.

A typical object in OpenText eDirectory is 3 to 5 KB in size. Using this object size, you can quickly calculate disk space requirements for the number of objects you have or need. Keep in mind that the object size will grow depending upon how many attributes are completed with data and what the data is. If objects will hold binary large object (BLOB) data such as pictures, sounds, or biometrics, the object size will subsequently grow.

The larger the partitions, the slower the replication cycles. If you are using products that require the use of OpenText eDirectory, such as ZENworks and DNS/DHCP services, the OpenText eDirectory objects created by these products will affect the size of the containers they are located in. You might consider placing objects that are for administration purposes only, such as DNS/DHCP, in their own partition so user access is not affected with slower replication. Also, managing partitions and replicas will be easier.

If you are interested, you can easily determine the size of your OpenText eDirectory database or the Directory Information Base (DIB) Set.

- For Windows, look at the DIB Set at `\novell\nds\dibfiles`.
- For Linux, look at the DIB Set in the directory you specified during installation.

Deciding Which Containers to Create

In general, create containers for objects that have access needs in common with other OpenText eDirectory objects. This lets you service many users with one trustee assignment or login script. You can create containers specifically to make container login scripts more effective, or you can place two departments in one container to make login script maintenance more feasible.

Keep users close to the resources they need to limit traffic over the network. For example, people who work in the same department generally work near each other. They usually need access to the same file system and they print to the same printers.

Exceptions to general workgroup boundaries are not hard to manage. If two workgroups use a common printer, for instance, you can create an Alias object to the printer in one of the workgroups. You can create Group objects to manage some User objects within a workgroup or User objects across multiple workgroups. You can create Profile objects for subsets of users with unique login script requirements.

Guidelines for Partitioning Your Tree

When you partition OpenText eDirectory, you allow parts of the database to exist on several servers. With this capability, you can optimize network use by distributing the OpenText eDirectory data processing and storage load over multiple servers on the network. By default, a single partition is created. For more information on partitions, refer to "Partitions" on page 53. For information on creating partitions, refer to Chapter 6, "Managing Partitions and Replicas," on page 143.

The following are guidelines for most networks. However, depending on the specific configuration, hardware, and traffic throughput of the network, you might need to adjust some guidelines to fit your needs.

Determining Partitions for the Upper Layers of the Tree

Just as you design your tree with a pyramid design, you will also partition with a pyramid design. Your partition structure will have few partitions at the top of the tree and more partitions as you move toward the bottom. Such a design creates fewer subordinate references than an OpenText eDirectory tree structure that has more partitions at the top than at the bottom.

This pyramid design can be achieved if you always create the partitions relatively close to the leaf objects, particularly the users.

NOTE: An exception is the partition created at the root of the tree during installation.

When designing the partitions for the upper layers, keep the following in mind:

- Partition the top of the tree based on the WAN infrastructure. Place fewer partitions at the top of the tree with more at the bottom.

You can create containers for each site separated by WAN links (placing each Server object in its local container), then create a partition for each site.

- In a network with WAN links, partitions should not span multiple locations.

  This design ensures that replication traffic between different sites is not unnecessarily consuming WAN bandwidth.

- Partition locally around the servers. Keep physically distant servers in separate partitions.

Determining Partitions for the Lower Layers of the Tree

When designing the partitions for the lower layers of the OpenText eDirectory tree, keep the following in mind:

- Define lower-layer partitions by organizational divisions, departments, and workgroups, and their associated resources.
- Partition so that all objects in each partition are at a single location. This ensures that updates to OpenText eDirectory can occur on a local server.

Determining Partition Size

With OpenText eDirectory, we recommend the following design limits for partition sizes:

Element	Limit
Partition Size	Unlimited Objects
	Replica Directory Information Base (DIB) limited to ITB
Total number of partitions in tree	Unlimited
Number of child partitions per parent	150
Number of replicas per partition	50
	Limited by replica DIB
Number of replicas per replica server	250

This change in design guidelines from NDS® 6 and 7 is due to architectural changes in NDS 8. These recommendations apply to distributed environments such as corporate enterprises. These recommendations might not subsequently apply to e-business or applications.

Although typical e-business users require that all the data be stored on a single server, OpenText eDirectory provides filtered replicas that can contain a subset of objects and attributes from different areas of the tree. This allows for the same e-business needs without storing all the data on the server. For more information, see “Filtered Replicas” on page 60.

Considering Network Variables

Consider the following network variables and their limitations when planning your partitions:

- The number and speed of servers

- The speed of network infrastructure (such as network adapters, hubs, and routers)
- The amount of network traffic

Guidelines for Replicating Your Tree

Creating multiple OpenText eDirectory partitions does not, by itself, increase fault tolerance or improve performance of the directory. However, strategically using multiple replicas does. The placement of replicas is extremely important for accessibility and fault tolerance. OpenText eDirectory data needs to be available as quickly as possible and needs to be copied in several places to ensure fault tolerance. For information on creating replicas, refer to Chapter 6, “Managing Partitions and Replicas,” on page 143.

The following guidelines will help determine your replica placement strategy.

- “Workgroup Needs” on page 84
- “Fault Tolerance” on page 84
- “Determining the Number of Replicas” on page 85
- “Replicating the Tree Partition” on page 85
- “Replicating for Administration” on page 86
- “Managing WAN Traffic” on page 86

Workgroup Needs

Place replicas of each partition on servers that are physically close to the workgroup that uses the information in that partition. If users on one side of a WAN link often access a replica stored on a server on the other side, place a replica on servers on both sides of the WAN link.

Place replicas in the location of highest access by users, groups, and services. If groups of users in two separate containers need access to the same object within another partition boundary, place the replica on a server that exists in the container one level above the two containers holding the group.

Fault Tolerance

If a disk crashes or a server goes down, replicas on servers in other locations can still authenticate users to the network and provide information on objects in partitions stored on the disabled server.

With the same information distributed on several servers, you are not dependent on any single server to authenticate you to the network or to provide services (such as login).

To create fault tolerance, plan for three replicas for each partition if the directory tree has enough servers to support that number. There should be at least two local replicas of the local partition. There is no need to have more than three replicas unless you need to provide for accessibility of the data at other locations, or you participate in e-business or other applications that need to have multiple instances of the data for load balancing and fault tolerance.

You can have only one master replica. Additional replicas must be read/write, read-only, or filtered. Most replicas should be read/write. They can handle object viewing, object management, and user login, just as the master replica can. They send out information for synchronization when a change is made.

Read-only replicas cannot be written to. They allow object searching and viewing, and they are updated when the replicas of the partition synchronize.

Do not depend on a subordinate reference or filtered replicas for fault tolerance. A subordinate reference is a pointer and does not contain objects other than the partition root object. Filtered replicas do not contain all objects within the partition.

OpenText eDirectory allows for an unlimited number of replicas per partition, but the amount of network traffic increases as the number of replicas increase. Balance fault tolerance needs with network performance needs.

You can store only one replica per partition on a server. A single server can store replicas of multiple partitions.

Depending on your organization's disaster recovery plan, the major work of rebuilding the network after a loss of a server or location can be done using partition replicas. If the location has only one server, back up OpenText eDirectory regularly. Consider purchasing another server for fault tolerance replication.

NOTE: ◆Some backup software does not back up OpenText eDirectory automatically.

- We recommend you exclude the DIB directory on your OpenText eDirectory server from any antivirus or backup software processes. Use the OpenText eDirectory Backup Tool to back up your DIB directory. For more information about backing up OpenText eDirectory, see “Backing Up and Restoring OpenText eDirectory” on page 417.

Determining the Number of Replicas

The limiting factor in creating multiple replicas is the amount of processing time and traffic required to synchronize them. When a change is made to an object, that change is communicated to all replicas in the replica ring. The more replicas in a replica ring, the more communication is required to synchronize changes. If replicas must synchronize across a WAN link, the time cost of synchronization is greater.

If you plan partitions for many geographical sites, some servers will receive numerous subordinate reference replicas. OpenText eDirectory can distribute these subordinate references among more servers if you create regional partitions.

Replicating the Tree Partition

The Tree partition is the most important partition of the OpenText eDirectory tree. If the only replica of this partition becomes corrupted, users will experience impaired functionality on the network until the partition is repaired or the OpenText eDirectory tree is completely rebuilt. You will also not be able to make any design changes involving the Tree.

When creating replicas of the Tree partition, balance the cost of synchronizing subordinate references with the number of replicas of the Tree partition.

Replicating for Administration

Because partition changes originate only at the master replica, place master replicas on servers near the network administrator in a central location. It might seem logical to keep masters at remote sites. However, master replicas should be where the partition operations will occur.

We recommend that major OpenText eDirectory operations, such as partitioning, be handled by one person or group in a central location. This methodology limits errors that could have adverse effects to OpenText eDirectory operations and provides for a central backup of the master replicas.

The network administrator should perform high-cost activities, such as creating a replica, at times when network traffic is low.

Managing WAN Traffic

If users currently use a WAN link to access particular directory information, you can decrease access time and WAN traffic by placing a replica containing the needed information on a server that users can access locally.

If you are replicating the master replicas to a remote site or are forced to place replicas over the WAN for accessibility or fault tolerance, keep in mind the bandwidth that will be used for replication.

Replicas should only be placed in non-local sites to ensure fault tolerance if you are not able to get the recommended three replicas, increase accessibility, and provide centralized management and storage of master replicas.

Planning the User Environment

After you have designed the basic structure of the OpenText eDirectory tree and have set up partitioning and replication, you should plan the user environment to simplify management and increase access to network resources. To create a user environment plan, review the users' needs and create accessibility guidelines for each area.

Reviewing Users' Needs

When you review users' needs, consider the following:

- Physical network needs, such as printers or file storage space

  Evaluate if resources are shared by groups of users within a tree or shared by groups of users from multiple containers. Also consider the physical resource needs of remote users.
- Bindery services needs for users

  Consider which applications are bindery-based and who uses them.
- Application needs

  Consider which applications and data files are needed by users, what operating systems exist, and which groups or users need access to applications. Consider if the shared applications should be manually or automatically launched by applications such as ZENworks.

Creating Accessibility Guidelines

After you have gathered information about user needs, you should determine the OpenText eDirectory objects that you will use to create the users' environments. For example, if you create policy packages or Application objects, you should determine how many you will create and where you will allow them to be placed in the tree.

You should also determine how you will implement security to restrict user access. You should identify any security precautions related to specific security practices. For example, you could warn network administrators to avoid granting the OpenText eDirectory Supervisor right to Server objects because this right is inherited by the file system.

Designing OpenText eDirectory for e-Business

If you use OpenText eDirectory for e-Business, whether you are providing a portal for services or sharing data with another business, the recommendations already mentioned in this chapter might not apply to you.

You might want to follow these suggested OpenText eDirectory e-business design guidelines instead:

- Create a tree with a limited number of containers.

  This guideline depends on the applications you use and your implementation of OpenText eDirectory. For example, a global deployment of a messaging server might require the more traditional OpenText eDirectory design guidelines discussed earlier in this chapter. Or, if you are going to distribute administration of users, you might create a separate Organizational Unit (OU) for each area of administrative responsibility.

- Maintain at least two partitions.

  Maintain the default partition at the Tree level, and create a partition for the rest of the tree. If you have created separate OUs for administrative purposes, create partitions for each of the OUs.

  If you are splitting the load over multiple servers, consider limiting the number of partitions, but still maintain at least two for backup or disaster recovery.

- Create at least three replicas of your tree for fault tolerance and load balancing.

  Keep in mind that LDAP does not load balance itself. To balance the load on LDAP, consider using Layer 4 switches.

- Create a separate tree for e-Business. Limit the network resources, such as servers and printers, included in the tree. Consider creating a tree that contains only User objects.

  You can use OpenText Identity Manager to link this user tree to your other trees that contain network information. For more information, see the (https://www.netiq.com/documentation/identity-manager-4.10/).

- Use auxiliary classes to customize your schema.

  If a customer or application requires a User object that is different from the standard inetOrgPerson, use auxiliary classes to customize your schema. Using auxiliary classes allows application designers to change the attributes used in the class without needing to re-create the tree.

- Increase LDIF-import performance.

When the OpenText Import Conversion Export utility is used, OpenText eDirectory indexes each object during the process. This can slow down the LDIF-import process. To increase the LDIF-import performance, suspend all indexes from the attributes of the objects you are creating, use the OpenText Import Conversion Export utility, then resume indexing the attributes.

- Implement globally unique common names (CN).

  OpenText eDirectory allows the same CN in different containers. However, if you use globally unique CNs, you can perform searches on CN without implementing logic for dealing with multiple replies.

Understanding the OpenText Certificate Server

OpenText Certificate Server allows you to mint, issue, and manage digital certificates by creating a Security container object and an Organizational Certificate Authority (CA) object. The Organizational CA object enables secure data transmissions and is required for Web-related products. The first OpenText eDirectory SP4 server will automatically create and physically store the Security container object and Organizational CA object for the entire OpenText eDirectory tree. Both objects are created and must remain at the top of the OpenText eDirectory tree.

Only one Organizational CA object can exist in an OpenText eDirectory tree. After the Organizational CA object is created on a server, it cannot be moved to another server. Deleting and re-creating an Organizational CA object invalidates any certificates associated with the Organizational CA.

IMPORTANT: Make sure that the first OpenText eDirectory server is the server that you intend to permanently host the Organizational CA object and that the server will be a reliable, accessible, and continuing part of your network.

If this is not the first OpenText eDirectory server on the network, the installation program finds and references the OpenText eDirectory server that holds the Organizational CA object. The installation program accesses the Security container and creates a Server Certificate object.

If an Organizational CA object is not available on the network, Web-related products will not function.

Rights Required to Perform Tasks on OpenText Certificate Server

To complete the tasks associated with setting up OpenText Certificate Server, the administrator needs to have rights as described in the following table.

OpenText Certificate Server Task	Rights Required
Base security setup for installing the first server into a new tree or upgrading the first server in a tree where there is no base security previously installed	Supervisor right at the root of the tree Supervisor right on the Security container
Base security setup for installing subsequent servers	Supervisor right on the server's container Supervisor right on the W0 object (located inside the Security container)
Creating the Organizational CA	Supervisor right on the Security container
Creating Server Certificate objects	Supervisor right on the server's container Read right to the NDSPKI:Private Key attribute on the Organizational CA's object

The root administrator can also delegate the authority to use the Organizational CA by assigning the following rights to subcontainer administrators. Subcontainer administrators require the following rights to install OpenText eDirectory with SSL security:

- Read right to the NDSPKI:Private Key attribute on the Organizational CA's object, located in the Security container.
- Supervisor right to the W0 object located in the Security container, inside the KAP object.

These rights are assigned to a group or a role, where all the administrative users are defined. For a complete list of required rights to perform specific tasks associated with OpenText Certificate Server, see Chapter 25, "Understanding the Certificate Server," on page 633.

Ensuring Secure OpenText eDirectory Operations on Linux Computers

eDirectory includes Public Key Cryptography Services (PKCS), which contains the OpenText Certificate Server that provides Public Key Infrastructure (PKI) services, Novell International Cryptographic Infrastructure (NICI), and SAS-SSL server.

The following sections provide information about performing secure OpenText eDirectory operations:

- "Verifying Whether NICI Is Installed and Initialized on the Server" on page 90
- "Initializing the NICI Module on the Server" on page 90
- "Starting the Certificate Server (PKI Services)" on page 90
- "Stopping the Certificate Server (PKI Services)" on page 90
- "Creating an Organizational Certificate Authority Object" on page 90
- "Creating a Server Certificate Object" on page 91
- "Exporting an Organizational CA's Self-Signed Certificate" on page 91

For information about using external certificate authority, see Chapter 25, "Understanding the Certificate Server," on page 633.

Verifying Whether NICI Is Installed and Initialized on the Server

Verify the following conditions, which indicate that the NICI module has been properly installed and initialized:

- The file `/etc/nici.cfg` exists
- The directory `/var/novell/nici` exists
- The file `/var/novell/nici/primenici` exists

If these conditions are not met, follow the procedure in the next section, "Initializing the NICI Module on the Server" on page 90.

Initializing the NICI Module on the Server

1. Stop the OpenText eDirectory server.
   - On Linux systems, enter

     ```
     /etc/init.d/ndsd stop
     ```

   IMPORTANT: We recommend you to use ndsmanage to start and stop ndsd.
2. Verify whether the NICI package is installed.
   - On Linux systems, enter

     ```
     rpm -qa | grep nici
     ```
3. (Conditional) If the NICI package is not installed, install it now.

   You will not be able to proceed if the NICI package is not installed.
4. Start the OpenText eDirectory server.
   - On Linux systems, enter:

     ```
     /etc/init.d/ndsd start
     ```

   IMPORTANT: We recommend you to use ndsmanage to start and stop ndsd.

Starting the Certificate Server (PKI Services)

To start PKI services, enter:

```
npki -1
```

Stopping the Certificate Server (PKI Services)

To stop PKI services, enter:

```
npki -u
```

Creating an Organizational Certificate Authority Object

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Creating an Organizational Certificate Authority Object" (https://www.netiq.com/documentation/edir88/crtadmin88/data/fbgccghh.html) in the *OpenText Certificate Server 3.3 Administration Guide*.

3 On the OpenText Identity Console home page > click **Certificate Management** tile.

4 Click **CA Management** > **Certificate Authority Management**.

If no Organizational Certificate Authority object exists, this opens the Create an Organizational Certificate Authority Object dialog box and the corresponding wizard that creates the object. Follow the prompts to create the object. For specific information on the dialog box or any of the wizard pages, click **Help**.

NOTE: You can have only one Organizational CA for your OpenText eDirectory tree. For more information about creating an Organizational CA, see "Create an Organizational Certificate Authority for Your Organization" on page 636.

Creating a Server Certificate Object

Server Certificate objects are created in the container that holds the OpenText eDirectory Server object. Depending on your needs, you might create a separate Server Certificate object for each cryptography-enabled application on the server. Or you might create one Server Certificate object for all applications used on that server.

NOTE: The terms Server Certificate Object and Key Material Object (KMO) are synonymous. The schema name of the OpenText eDirectory object is NDSPKI:Key Material.

---

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Creating a Server Certificate Object" on page 647.

3 On the OpenText Identity Console home page > click **Certificate Management** tile.

4 Click **Server Certificate Management** > click **Create Server Certificate** +.

This opens the Create Server Certificate Wizard. Follow the prompts to create the object. For specific information on any of the wizard pages, click **Help**.

Exporting an Organizational CA's Self-Signed Certificate

A self-signed certificate can be used for verifying the identity of the Organizational CA and the validity of a certificate signed by the Organizational CA.

From the Organizational CA's property page, you can view the certificates and properties associated with this object. From the Self-Signed Certificate property page, you can export the self-signed certificate to a file for use in cryptography-enabled applications.

The self-signed certificate that resides in the Organizational CA is the same as the Trusted Root certificate in a Server Certificate object that has a certificate signed by the Organizational CA. Any service that recognizes the Organizational CA's self-signed certificate as a trusted root will accept a valid user or server certificate signed by the Organizational CA.

1. On the OpenText Identity Console home page, click **Certificate Management** tile > **CA Management**.
2. Click the **Certificates** tab, then click **Self-Signed Certificate**.
3. Click **Export**.

   This opens the Export Certificate Wizard. Follow the prompts to export the certificate. For specific information on any of the wizard pages, click **Help**.
4. On the Export Certificate Summary page, click **Save the Exported Certificate to a File**.

   The certificate is saved to a file and is available to be imported into a cryptography-enabled application as the trusted root.
5. Click **Close**.

Include this file in all command line operations that establish secure connections to OpenText eDirectory.

Synchronizing Network Time

Time synchronization is a service that maintains consistent server time across the network. Time synchronization is provided by the server operating system, not by OpenText eDirectory. OpenText eDirectory maintains its own internal time to ensure the proper order of OpenText eDirectory packets, but it gets its time from the server operating system.

If your network uses Windows or Linux, you should use Network Time Protocol (NTP) to synchronize the servers, because it is a widely-used standard to provide time synchronization.

NTP

NTP functions as part of the UDP protocol suite, which is part of the TCP/IP protocol suite. Therefore, a computer using NTP must have the TCP/IP protocol suite loaded. Any computers on your network with Internet access can get time from NTP servers on the Internet.

NTP synchronizes clocks to the Universal Time Coordinated (UTC) standard, which is the international time standard.

NTP introduces the concept of a stratum. A stratum-1 server has an attached accurate time piece such as a radio clock or an atomic clock. A stratum-2 server gets time from a stratum-1 server, and so on.

For more information on time synchronization software, see The Network Time Protocol (http://www.ntp.org) Web site.

Synchronizing Time on Linux Computers

You can use the xntpd Network Time Protocol (NTP) daemon to synchronize time on Linux servers. xntpd is an operating system daemon that sets and maintains the system time-of-day in synchronism with Internet standard time servers.

For information on running ntpd on Linux systems, see ntpd - Network Time Protocol (NTP) Daemon (http://www.eecis.udel.edu/~mills/ntp/html/ntpd.html).

Verifying Time Synchronization

To verify that time is synchronized in the tree, run DSRepair from a server in the Tree that has at least Read/Write rights to the Tree object.

Windows

1. Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**.
2. Click **dsrepair.dlm** > **Start**.
3. Click **Repair** > **Time Synchronization**.

Linux

1. Run the following command:

   ```
   ndsrepair -T
   ```

3 Managing Objects

OpenText eDirectory includes OpenText Identity Console, a Web-based network management application that lets you manage the objects in your OpenText eDirectory tree. To understand the features and benefits of OpenText Identity Console, see the *OpenText Identity Console Administration Guide* (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

Managing OpenText eDirectory objects involves creating, modifying, and manipulating objects. For example, you might need to create user accounts and administer user rights. Use OpenText Identity Console to:

- Perform administration basics, such as browsing, creating, editing, and organizing objects.
- Create user accounts, including specifying a user's login name and supplying other information used by OpenText eDirectory
- Administer rights (assign rights, grant equivalence, block inheritance, and view effective rights). See “Administering Rights” on page 69 for more information.
- Configure role-based administration (define administrator roles for specific administrative applications through the role-based services object).

This chapter contains information on the following topics:

- “General Object Tasks” on page 95
- “Managing User Accounts” on page 99
- “Configuring Roles and Access Control” on page 106

General Object Tasks

This section contains steps for basic tasks you will use when managing your OpenText eDirectory tree:

- “Browsing the OpenText eDirectory Tree” on page 96
- “Creating an Object” on page 97
- “Modifying an Object's Properties” on page 98
- “Copying Objects” on page 98
- “Moving Objects” on page 98
- “Deleting Objects” on page 99
- “Renaming Objects” on page 99

Browsing the OpenText eDirectory Tree

The **Obejct Management** tile on OpenText Identity Console home page lets you search or browse for objects in your OpenText eDirectory tree. You can view the structure of your tree and perform tasks. The tasks available depend on the type of object you select.

The OpenText eDirectory **Object Management** tile in OpenText Identity Console also lets you search or browse for objects. In most entry fields in OpenText Identity Console, you can specify an object name and context, or you can select the **Type** to search or browse for the object you want. Selecting an object in the OpenText eDirectory **Object Management** tile inserts the object and the object's context into the entry field.

This section contains the following information:

- "Using the View Object Button" on page 96
- "Using the Object Selector Button" on page 97

Using the View Object Button

Use the techniques described below to locate the specific objects you want to manage.

- "Using Browse" on page 96
- "Using Search" on page 97

Using Browse

1. On OpenText Identity Console home page, click the **Object Management** tile.
2. On the **Objects** page, provide **Name**, select **Type**, provide or search for **Context**.
3. Use the following options to browse for an object:

Option	Description
Context	Lets you specify the name of the container whose contents you want to view. To use this option, click  and select the container you want.
Name	Lets you specify the name of an object. You can use an asterisk (*) as a wildcard character in this field. For example, `g*` finds all objects starting with "g," such as Germany or Greg, and `*te` finds all entries ending in "te," such as Kate or Corporate.
Type	Lets you specify the type of object you want to search for. The default is All Available Types. To use this option, select an object type from the drop-down list.

4. Click **Search**.

Using Search

1. On the OpenText Identity Console home page, click the **Object Management** tile.
2. On the **Objects** page, specify the **Name** of the object you want to search for.

   You can use an asterisk (*) as a wildcard character in this field. For example, `g*` finds all objects starting with “g,” such as Germany or Greg, and `*te` finds all entries ending in “te,” such as Kate or Corporate.
3. From the **Type** drop-down list select the type of object you want to search for.
4. In the **Context** field, specify the name of the container you want to search in.

   Click **Search Sub-containers** to include all subcontainers located within the current container in the search.
5. Click **Search**.
6. When you find the object you are looking for, right-click the object, then choose from the list of available tasks to perform.

Using the Object Selector Button

Use the techniques described below to locate the specific objects you want to manage.

- “Using Browse” on page 97
- “Using Search” on page 97

Using Browse

1. On the OpenText Identity Console home page> click **Tree View** tile.
2. Click the containers and users to perform required operations.

Using Search

1. On the OpenText Identity Console home page, click **Object Management** tile.

   Objects page opens.
2. Enter name in the Name field, you can use an asterisk (*) as a wildcard character in this field. For example, `g*` finds all objects starting with “g,” such as Germany or Greg, and `*te` finds all entries ending in “te,” such as Kate or Corporate.
3. Select the **Type** from drop-down list.
4. Select Context, specify the name of the container you want to search in.

   Click **Search Sub-containers** to include all subcontainers located within the current container in the search.
5. Click **Search**.

Creating an Object

1. On the OpenText Identity Console home page, click **Object Management** Tile .
2. Click **Create Object** .

3 Provide a Name > select a Type > click for Context > **Next**.

4 Click **Create**.

The information requested depends on the type of object you are creating.

Modifying an Object's Properties

1 On the OpenText Identity Console home page, click **Object Management** Tile .

2 Select the object that you want to modify, then click .

3 Specify the name and context of the object or objects you want to modify, then click **OK**.

4 Edit the property pages that you want.

5 Click **Save**.

Copying Objects

This option lets you create a new object with the same attribute values as an existing object, or copy attribute values from one object to another.

1 On the OpenText Identity Console home page, click **DN Management** tile.

2 Click **Move** > click **Search** to which you want to copy.

3 Click **Object** , specify the name and context of the object you want to copy.

4 Select one of the following options:

- **Create New Object and Copy Attribute Values**
- **Copy Attribute Values to an Existing Object**

5 If you want to copy access control list (ACL) rights to the object you are creating/modifying, select **Copy ACL Rights**.

Copying ACL rights can take additional processing time depending upon your system and networking environment.

6 Click **OK**.

Moving Objects

1 On the OpenText Identity Console home page, click the **DN Management** Tile .

2 Click **Move** > **Search** to specify the container you want to move the object or objects to > click **OK**.

3 In the **Object** field, specify the name and context of the object or objects you want to move.

4 In the **Move To** field, specify the container you want to move the object or objects to.

This allows any operations that are dependent on the old location to continue uninterrupted until you can update those operations to reflect the new location.

5 Click **Save**.

Deleting Objects

1 On the OpenText Identity Console home page, click **Object Management** Tile .

2 Select the object that you want to delete, then click .

Renaming Objects

1 On the OpenText Identity Console home page, click the **DN Management** Tile .

2 Click **Rename** > click **Search** to select the object or objects to rename.

3 In the **New Name** field, specify the name and context of the object you want to rename.

4 In the **New Object Name** field, specify the new name for the object.

Do not include the object's context in the **New Object Name** field.

5 If you want to create an Alias for the object being renamed, select **Create an Alias in Place of Renamed Object** .

This allows any operations that are dependent on the old object name to continue uninterrupted until you can update those operations to reflect the new name.

6 If you want to save the old object name, select **Save Old Name** .

This saves the old name as an additional (unofficial) value of the Name property. Saving the old name lets users search for the object based on that name. After renaming the object, you can view the old name in the **Other Name** field on the **General Identification** tab for that object.

7 Click **Save**.

Managing User Accounts

Setting up an OpenText eDirectory user account involves creating a User object and setting properties to control login and the user's network computing environment. You can use a template object to facilitate these tasks.

You can create login scripts to cause users to be connected automatically to the files, printers, and other network resources they need when they log in. If several users use the same resources, you can put the login script commands in container and profile login scripts.

This section contains the following information:

- "Creating and Modifying User Accounts" on page 100
- "Setting Up Optional Account Features" on page 101
- "Setting Up Login Scripts" on page 104
- "Login Time Restrictions for Remote Users" on page 105
- "Deleting User Accounts" on page 106

Creating and Modifying User Accounts

A user account is a User object in the OpenText eDirectory tree. A User object specifies a user's login name and supplies other information used by OpenText eDirectory to control the user's access to network resources.

This section contains the following information:

- "Creating a User Object" on page 100
- "Modifying a User Account" on page 100
- "Enabling a User Account" on page 100
- "Disabling a User Account" on page 101

Creating a User Object

1. On the OpenText Identity Console home page, click the **User Management** Tile.
2. Click **Add Users** +.
3. In the Add User page, provide the required user-related information:
   - **Username**
   - **Context**
   - **Last name**
   - **Password**
4. Specify a container to create the user in.
5. Specify any additional (optional) information you want, then click **Create**.

   User Created Successfully message appears.
6. Click **OK**.

Modifying a User Account

1. On the OpenText Identity Console home page, click the **User Management** Tile.
2. Select the **Users** > click **Modify User**.
3. On the **Modify User** page make changes that you want, then click **Save**.

Enabling a User Account

1. On the OpenText Identity Console home page, click the **Tree View** Tile.
2. Select **novell** > select the user > click **Action Items**.
3. On the **ACTION ITEMS** list, select **Enable Account**.
4. Click **OK**.

Disabling a User Account

1 On the OpenText Identity Console home page, click the **Tree View** Tile.

2 Select **novell** > select the user > click **Action Items**.

3 On the **ACTION ITEMS** list, select **Disable Account**.

4 Click **OK**.

Setting Up Optional Account Features

After creating a User object, you can set up the user's network computing environment and implement extra login security features.

Setting Up a User's Network Computing Environment

1 On the OpenText Identity Console home page, click the **User Management** Tile.

2 Select the User > click **Modify**.

3 On the **Modify User** page click **General** tab > **Environment and Login Script**.

4 Fill in the required script in the **Login script** field.

5 Click **Save**.

Setting Up Extra Login Security for a User

1 On the OpenText Identity Console home page, click the **User Management** Tile.

2 Select **Users** > **Modify User**.

3 On the **Modify User** page > **Restrictions** > **Intruder Lockout**.

4 In the **Intruder Lockout** field, fill in the property pages you want.

Click for details on any page.

Page	Description	LDAP Attribute
Password Restrictions	Sets up a login password.	passwordRequired
Login Restrictions	◆ Enable or disable the account. ◆ Limit the number of concurrent login sessions. ◆ Set a login expiration and lockout date.	loginDisabled loginMaximumSimultaneous loginExpirationTime or loginGraceLimit
Time Restrictions	Restricts the times when the user can be logged in. If you set a restriction and the object is logged in when the restricted time arrives, the system issues a five-minute warning and then (after five minutes) logs the object out if it isn't logged out already.If the user will log in remotely, see "Login Time Restrictions for Remote Users" on page 105.	loginAllowedTimeMap
Address Restrictions	Restricts the network locations (workstations) that this user can log in from. If you don't set restrictions on this page, the user can log in from any network location.	networkAddressRestriction
Account Balance	Sets up an accounting of this user's server usage.	accountBalance
Intruder Lockout	Lets you work with this account if it has been locked because of intruder detection. To manage the intruder detection setup, use the Intruder Detection property page of the parent container.	lockedByIntruder

5 Click **Save**.

Setting Up Intruder Detection for All Users in a Container

1 On the OpenText Identity Console home page, click the **User Management** Tile.

2 Click **Users** > **Modify User** .

3 On the **Modify User** page > **Restrictions** > **Intruder Lockout** > then click **OK**.

4 Enter the required parameters:

Option	Description
Detect Intruders	Enables the intruder detection system for the user accounts in the container.
Incorrect Login Attempts	Specifies the number of consecutive failed login attempts that are allowed before intruder detection is activated. If a person uses any of the user accounts in this container to log in and fails consecutively more than this number of times, intruder detection is activated. The number is stored in the Login Intruder Limit property of the container.
Intruder Attempt Reset Interval	Specifies the time span in which consecutive failed logins must occur for intruder detection to be activated. Enter the number of days, hours, and minutes.
Lock Account After Detection	Specifies whether to disable login if intruder detection is activated on a user account in this container. If you don't check this check box, no action is taken when intruder detection is activated. If you check this check box and the system locks a user account due to intruder detection, you can unlock the account by unchecking the Account Locked check box on the Intruder Lockout property page of the User object.
Days, Hours, Minutes	These three fields specify the length of time that login is disabled when intruder detection is activated on a user account in this container. Enter the number of days, hours, and minutes you want, or accept the default of 15 minutes. After the specified time elapses, the system re-enables login for the user account. The contents of these fields are stored in the Intruder Lockout Reset Interval property of the container. If the values of these three fields are specified as zero then the user account is locked indefinitely.

5 Click **Save**.

Disabling the Login Time Update Interval

You can specify an interval value to disable the update of the login time attribute of a user. You can specify the interval value for a user, container, and Login Policy. Security object (LPO), or server. To enable this feature, the schema needs to be extended using the `nmas.sch` file.

To specify the interval for a user:

1 On the OpenText Identity Console home page, click **User Management**.

2 Select a user for which the login time interval has to be updated.

3 Click Modify User > go to **Others** tab.

4 Click **Valued Attributes** , and then select **sasUpdateLoginTimeInterval** from **Select Attributes** list.

5 Click **OK** and specify the login time interval and other required information.

6 Click **Save** and click **OK**.

To specify the update interval for container and LPO:

1. On the OpenText Identity Console home page, click **Object Management** > on the **Type** drop down, select **Organization Unit** > click **Search**.
2. Specify the name and context of a container or login policy object.
3. Click **Organization Unit**.
4. On the **General** tab > click **Add Optional Attributes** > select **sasUpdateLoginTimeInterval** > click **OK**.
5. Use the arrow button to move **sasUpdateLoginTimeInterval** from unValued Attributes list to the **Valued Attributes** list, as necessary, then click **Save**.

Setting Up Login Scripts

A login script is a list of commands that executes when a user logs in. It is typically used to connect the user to network resources like files and printers. Login scripts execute on the user's workstation in the following order:

1. Container login script
2. Profile login script
3. User login script

During login, if the system doesn't find one of these login scripts, it skips to the next one in the list. If none are found, the system executes a default script that maps a search drive to a folder on the user's default server. The default server is set on the Environment property page of the user object.

Creating a Login Script

1. On the OpenText Identity Console home page, click the **User Management** tile > select the User > click **Modify** .
2. On the **Modify User** page > **General** > **Environment and Login Script**.
3. Click **Environment and Login Script** drop down, at the **Login script** field, enter the login script command.

   See the *Login Scripts Guide* (http://www.novell.com/documentation/linux_client/login/data/front.html) for more information.

To Have the Login Script Apply To	Create It On
One user only	The User object
One or more users that haven't been created yet	A Template object
All the users in a container	The container object
A set of users in one or more containers	A Profile object

4. Click **Save** > **OK**.

Assigning a Profile to a User

Associating a profile with a User object causes the profile's login script to execute during the user's login. Make sure that the user has Browse rights to the Profile object and Read rights to the Login Script property of the profile object.

See "Viewing Effective Rights to an OpenText eDirectory Object or Property" on page 73 for more information.

1. **On the OpenText Identity Console home page, click the User Management** tile > select the User > click **Modify** .
2. On the **Modify User** page > **General** > **Environment and Login Script** > **Profile** .
3. The CONTEXT BROWSER page > search the User object that you want to create the login script on.
4. Click **Save**.

Login Time Restrictions for Remote Users

On the Time Restrictions property page of a User object, you can restrict the times when the user can be logged in to OpenText eDirectory. By default, there are no login time restrictions.

If you set a login time restriction and the user is logged in when the restricted time arrives, the system issues a warning to log out within five minutes. If the user is still logged in after five minutes, he or she is logged out automatically and loses any unsaved work.

If a user logs in remotely from a different time zone than the server processing the login request, any login time restrictions that have been set for the user are adjusted for the time difference. For example, if you restrict a user from logging in Mondays from 1:00 a.m. to 6:00 a.m. and the user logs in remotely from a time zone that is one hour later than the server, the restriction effectively becomes 2:00 a.m. to 7:00 a.m. for that user.

1. On the OpenText Identity Console home page, click, **Use Management** tile.
2. Select **Users**.
3. Click **Restrictions** tab, click **Time Restrictions**.
4. Select from the following options:

Option	Description
Time Grid	Each cell in the time grid represents a half hour on a particular day of the week. Red cells represent restricted times (when this object cannot be logged in). Gray cells represent unrestricted times (when the object can be logged in). To create a time restriction, click the desired times to make them dark gray. You can also select multiple times by holding down the Shift key, clicking a cell, then dragging across the corresponding cells. The login time restrictions you set are stored in the Login Allowed Time Map property of this object.
Add Time Restrictions	To add a time restriction, select a gray cell, then select this option.
Remove Time Restrictions	To remove a time restriction, select a red cell, then select this option.
Update	Click this button to enable the selection.
Reset	Click this button to reset the time grid to the way it was before you opened this property page.

5 Click **Save**.

Deleting User Accounts

1 On the OpenText Identity Console home page, click the **User Management** tile > select the User.

2 Click **Delete User** .

The User is deleted.

Configuring Roles and Access Control

OpenText Identity Console gives administrators the ability to assign specific responsibilities to users and to present the user with only the tools (and their accompanying rights) necessary to perform those sets of responsibilities. This functionality is called *Roles and Access Control (RAC).*

RAC allows administrators to focus the user on a specified set of functions, called *Tasks*, and objects as determined by the grouping of tasks called *Roles*. What users see when they access OpenText Identity Console is based on their role assignments in OpenText eDirectory. Only the tasks assigned OpenText to that user are displayed. The user does not need to browse the tree to find an object to administer. The OpenText Identity Console tile for that task presents the necessary tools and interface to perform the task.

You can assign multiple roles to a single user. You can also assign the same role to multiple users.

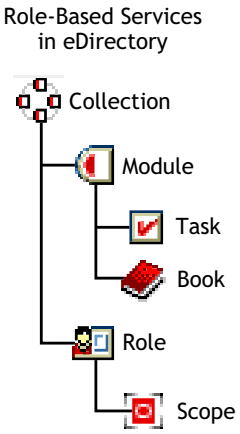
RAC is represented by objects defined in OpenText eDirectory. The base OpenText eDirectory schema gets extended during the OpenText Identity Console installation. The RAC object types are listed in the following table.

Object	Description
RAC Configuration	A container object that holds all RAC Role and Module objects. RAC Configuration objects are the topmost containers for all RAC objects. A tree can have any number of RAC Configuration objects. These objects have “owners,” which are users who have management rights over the RAC Configuration. RAC Configuration objects can be created in any of the following containers: ◆ Domain ◆ Location ◆ Country ◆ Organization ◆ Organizational Unit
RAC Role	A container object that specifies the tasks that users (members) are authorized to perform. Defining a Role includes creating an RAC Role object and specifying the tasks that the role can perform. Role members can be Users, Groups, Organizations, or Organizational Units, and they are associated to a role in a specific scope of the tree. The RAC Task and RAC Book objects are assigned to RAC Role objects. RAC Role objects can be created only in RAC configuration containers.
RAC Module	A container object that holds RAC Task and RAC Book objects. RAC Module objects have a module name attribute that represents the name of the product that defines the tasks or books (for example, OpenText eDirectory Maintenance, NMAS, or OpenText Certificate Access). RAC Module objects can be created only in RAC Configuration containers.
RAC Task	A leaf object that represents a specific function, such as resetting login passwords. RAC Task objects are located only in RAC Module containers.
RAC Book	A leaf object that containing a list of pages assigned to the book. An rbsBook can be assigned to one or more Roles and to one or more Object class types. rbsBook objects are located only in rbsModule containers.

Object	Description
RAC Scope	A leaf object used for ACL assignments (instead of making assignments for each User object). RAC Scope objects represent the context in the tree where a role will be performed and are associated with RAC Role objects. They inherit from the Group class. User objects are assigned to an RAC Scope object. These objects have a reference to the scope of the tree that they are associated with. This object is dynamically created when needed, then automatically deleted when no longer needed. They are located only in RAC Role containers. **WARNING:** Never change the configuration of a Scope object. Doing so will have serious consequences and could possibly break the system.

The RAC objects reside in the OpenText eDirectory tree as depicted in the following figure.

Figure 3-1 *RAC Objects in the OpenText eDirectory Tree*



Defining RAC Roles

RAC roles specify the tasks that users are authorized to perform. Defining an RAC role includes creating an RAC Role object and specifying the tasks that the role can perform and the User, Group, or container objects that can perform those tasks. In some cases, OpenText Identity Console Tiles (product packages) provide predefined RAC roles that you can modify.

The tasks that RAC roles can perform are exposed as RAC Task objects in your OpenText eDirectory tree. These objects are added automatically during the installation of product packages. They are organized into one or more RAC Modules, which are containers that correspond to the different functional modules of the product.

For information on assigning members to a role, see “Assigning RAC Role Membership and Scope” on page 109.

- “Creating a Role Object” on page 109
- “Modifying the Tasks Associated with a Role” on page 109
- “Assigning RAC Role Membership and Scope” on page 109
- “Deleting a Role in RAC Configuration” on page 110

Creating a Role Object

Use the RAC Configuration (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/b8qqsec.html) to create a new RAC Role object. We recommend creating the new RAC Role object in the same RAC Configuration container where the other RAC Role objects reside (for example, the RAC Configuration container).

1. On the OpenText Identity Console home page, click **Roles and Access Control**.
2. In the **Roles** tab, click **Create Role** 
3. On the **Create Role** window, click **Add Scope** , and select a **Scope**.
    1. Enter the **Name**.
    2. Select the **Member** or the group of members that you want to associate with the Role.
4. From the **Select Task** menu, select the tasks that need to be assigned to the Role.
5. Click **Create**.

    A new Role is created. Close the window and open the Configuration again to view the changes.

See "Defining Custom RAC Tasks" on page 110 for information on adding members to roles.

Modifying the Tasks Associated with a Role

Each RAC role has a set of available tasks associated with it. You can choose which tasks are assigned to a particular role, adding or removing tasks as necessary.

1. On the OpenText Identity Console home page, click **Roles and Access Control** tile.
2. On the **RAC Configuration** tab click the configuration in which you want to modify a role
3. On the **Role** tab, click the role you want to modify.
4. (Optional) If you want to add tasks to a role, complete the following steps:
    - **4a** Click **Add/Remove Task** .
    - **4b** Select the Task, click **Confirm**.
5. (Optional) If you want to remove tasks from a role, complete the following steps:
    - **5a** Select the tasks you want to remove.
    - **5b** Click **Add/Remove Task** .
6. Click **Confirm**.

Assigning RAC Role Membership and Scope

After you have defined the RAC roles needed in your organization, you can assign members to each role. In doing so, you specify the scope in which each member can exercise the functions of the role. The scope is the location or context in the OpenText eDirectory tree where this role can be performed.

A user can be assigned to a role in the following ways:

- Directly

- Through group and dynamic group assignments. If a user is a member of a group or a dynamic group that is assigned to a role, then the user has access to the role.
- Through organizational role assignments. If a user is an occupant of a organizational role that is assigned a role, then the user has access to the role.
- Through container assignment. A user object has access to all of the roles that its parent container is assigned. This could also include other containers up to the root of the tree.

A user can be associated with a role multiple times, each with a different scope. You can also assign the same task to multiple members.

To assign role to an existing member:

1. On the OpenText Identity Console home page, click **Roles and Access Control** tile.
2. Click **RAC Configuration** tab.
3. Click **Edit Member Associations** + > select the member.
4. On the **Add Roles** window, select the role that is required.
   - **Add Role:** Specify, or use the Object Selector to find the desired object to be a role member.
   - **Add Scope:** Specify, or use the Object Selector to find the scope within which this member can perform the role.
5. Click **OK**.

   The roles are added successfully. Close the window and open the Configuration again to view the changes. For more information click Help.

Deleting a Role in RAC Configuration

1. On the OpenText Identity Console home page, click **Roles and Access Control** tile.
2. Click the **RAC** tab.
3. Click the configuration in which you want to delete an RAC role.
4. Select the role you want to modify.
5. Click **Remove Role**.
6. Click **OK**.

Defining Custom RAC Tasks

- "Creating an OpenText Identity Console Task" on page 110
- "Modifying the Associated Role" on page 111
- "Deleting a Task" on page 111

Creating an OpenText Identity Console Task

Create an OpenText Identity Console task through **Custom Forms**, or through **External Applications**. The forms created here reflects as a Task. For information see: Managing Custom Forms (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/alw1iwfbb2eer.html) and Managing External Application (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/alw1iwfbb2eerbb3b.html).

Modifying the Associated Role

1. On the OpenText Identity Console home page, click **Roles and Access Control** tile.
2. Click **Edit Member Association** tab > **Select Member** .
3. The list of associations that the user is already part of are displayed.
4. Modify the **Scope**, **Assigned Rights**, and **Inheritable** object as required.
5. Click **OK**, and close.

Deleting a Task

1. On the OpenText Identity Console home page, click **Roles and Access Control** tile.
2. Click **RAC Configuration** tab.
3. Click the configuration in which you want to delete a task.
4. Select the task you want to delete.
5. Click **Delete** .
6. Click **OK**.

4 Managing Background Process

To cater to large dynamic environments, OpenText eDirectory provides optimized background processes and configuration options to tune your systems appropriate to your environment.

This chapter includes the following topics:

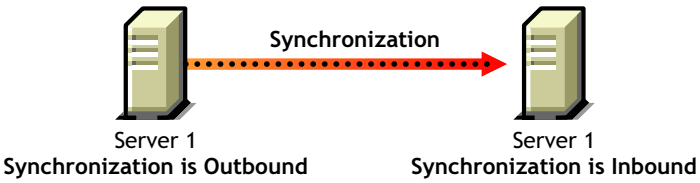
- "Synchronization" on page 113
- "Configuring Background Processes" on page 128

Synchronization

Synchronization is the transfer of directory information from one replica to another, so the information in each partition is consistent with the other. OpenText eDirectory automatically keeps the servers in the replica ring synchronized.

Synchronization consists of inbound and outbound synchronization. For example, if the modifications to data have to be synchronized from server1 and server2, the term *outbound* refers to the synchronization process that is sent from server1 to server2. The term *inbound* refers to the synchronization process that is received by server2 from server1.

Figure 4-1 *Outbound and Inbound Synchronization*



There are two types of synchronization:

- Normal Synchronization or Replica Synchronization
- Priority Sync

The following table gives you a comparison between normal synchronization and priority sync:

Table 4-1 *Comparison between Normal or Replica Synchronization and Priority Sync*

Normal Synchronization or Replica Synchronization	Priority Sync
Triggered when there are modifications to data in any of the servers in the replica ring.	Triggered when there are modifications only to the data that you identify as critical.
For more information, refer to "Normal or Replica Synchronization" on page 116.	For more information, refer to "Priority Sync" on page 118.

Normal Synchronization or Replica Synchronization	Priority Sync
After the data is modified, the changes are buffered. Normal synchronization starts after approximately 30 seconds from the time the modifications are saved.	The changes to the critical data are not buffered. Priority sync starts immediately after the data is modified.
The most important synchronization in OpenText eDirectory. It happens irrespective of whether the modifications are synchronized by priority sync or not.	Complementary to normal synchronization. Though the critical attributes are synchronized through priority sync, they are synchronized again through normal synchronization.
Can happen between OpenText eDirectory 8.8 servers or across servers hosting earlier versions of OpenText eDirectory.	Happens only between OpenText eDirectory 8.8 and later servers, holding the same partition.
Never fails due to its feature. For more information, refer to "Features of Synchronization" on page 114.	If priority sync fails, the modifications to the critical data are synchronized through normal synchronization. For more information, refer to "When Can Priority Sync Fail?" on page 124.

NOTE: The Priority sync information in available in the SYDL or Synchronization Details tags in ndstrace, dstrace, or iMonitor trace screens.

Features of Synchronization

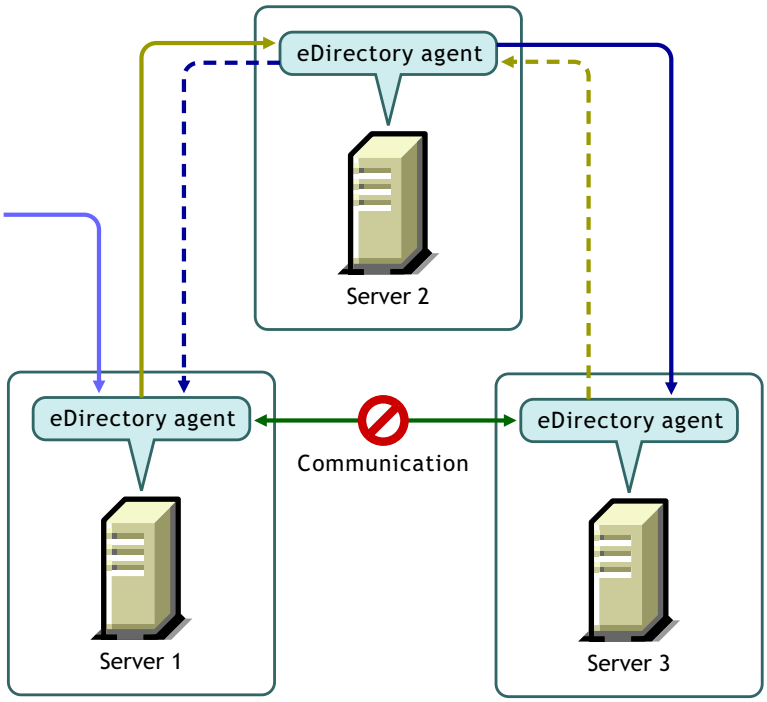
Synchronization in OpenText eDirectory

- Is transitive.
- Maintains object transaction model.
- Has timestamps like transitive vector, local received up to and remote received up to.

Transitive Synchronization

Synchronization in OpenText eDirectory is transitive. This means that OpenText eDirectory synchronizes the changes to the data without requiring the eDirectory agent to directly contact and synchronize those changes with every other agent in the replica ring.

Figure 4-2 *Transitive Synchronization*



For example, if you make a change to data on Server 1, the change is synchronized from Server 1 to Server 2 and from Server 2 to Server 3. Even if Server 1 could not come into direct contact with Server 3, because of a problem in communication, it still receives the latest change to the data, through Server 2. Server 3 lets Server 2 know that it has received the changes. Server 2 in turn tells Server 1 that Server 3 and itself are synchronized.

Object Transaction Model

Synchronization in OpenText eDirectory maintains the object transaction model, a standard for LDAP and X.500-compliant directories. Object transaction model means that all the previous transactions should be synchronized before synchronizing the new ones.

For example, you have modifications D1, D2, and D3 to the data on a server. Due to network failure, these modifications are not synchronized across other servers. If you make another modification D4 on the server, D4 will be synchronized only after D1, D2, and D3 are synchronized across all the servers in the replica ring.

Transitive Vector

A transitive vector is a time stamp for a replica. It is made up of a representation of the number of seconds since a common specific point in history (January 1, 1970), the replica number, and the current event number. Here's an example:s3D35F377 r02 e002

For more information, refer to “Transitive Vectors and the Restore Verification Process” on page 431.

Local Received Up To

Local Received Up To (LRUT) is the time before which the local replica has received the changes.

For more information, refer to “Browsing Objects in Your Tree” on page 237.

Remote Received Up To

Remote Received Up To (RRUT) is the LRUT of the remote replica.

For more information, refer to “Browsing Objects in Your Tree” on page 237.

Normal or Replica Synchronization

Normal or Replica Synchronization is one of the two synchronization processes in eDirectory. Normal synchronization synchronizes all the modifications to data on a server with other servers in the replica ring.

Normal synchronization happens across all servers having any version of eDirectory, having the same partition.

For more information, refer to “Administering Replicas” on page 147.

You can enable or disable normal synchronization by enabling or disabling outbound and inbound synchronization in OpenText iMonitor. Both inbound and outbound synchronizations are enabled by default. To sync the modifications to data across the other servers through normal synchronization, you need to configure the synchronization parameters in iMonitor. Refer to “Controlling and Configuring the DS Agent” on page 232 for more information.

In normal synchronization, when you make any modifications to the data, the changes you make are buffered before synchronizing them across the servers. You can view the synchronization status in the servers of your setup in iMonitor. Refer to “Browsing Objects in Your Tree” on page 237 for more information.

Normal synchronization maintains the object transaction model and is transitive. Refer to “Transitive Synchronization” and “Object Transaction Model” on page 101 for more information.

Configuring Normal Synchronization

You can configure normal synchronization using Agent Configuration under Agent Synchronization in iMonitor.

This section provides the following information:

- “Enabling/Disabling Normal Synchronization” on page 117
- “Enabling/Disabling Inline Cache” on page 117
- “Synchronization Threads” on page 117
- “Synchronization Method” on page 117

Enabling/Disabling Normal Synchronization

You can enable or disable normal synchronization by enabling or disabling the outbound and inbound synchronization in iMonitor. Refer to "Controlling and Configuring the DS Agent" on page 232 for more information.

Outbound synchronization is enabled by default. When you disable this option on a server, the modifications to the data on this server are not synchronized with other servers. You can specify the amount of time, in hours, for which you want the outbound synchronization disabled. The default which is also the maximum time is 24 hours. After the specified time, the modifications to the data on this server are synchronized with other servers.

Inbound synchronization is enabled by default. When you disable this option for a server, the modifications to the data on other servers are not synchronized with this server.

Enabling/Disabling Inline Cache

You can enable or disable the Inline Change Cache for a server. You can disable Inline Change Cache only when Outbound Synchronization is disabled. Enabling Outbound Synchronization also enables Inline Change Cache.

Disabling Inline Change Cache marks the change cache as invalid for this replica and tags it with an invalid flag in **Agent Configuration** > **Partitions**. Enabling Inline Change Cache removes the invalid change cache flag when the change cache is rebuilt.

Synchronization Threads

For outbound synchronization, you need to configure the synchronization threads. Using iMonitor, you can specify the number of synchronization threads using **Agent Configuration** under **Agent Synchronization**.The supported values are 1 to 16. See "Controlling and Configuring the DS Agent" on page 232 for more information.

Synchronization Method

Normally, OpenText eDirectory automatically chooses the method based on the number of replicas and replication partners. The following are the synchronization methods:

- **By Partition:** The modifications to data are synchronized simultaneously with other replicas.Several threads are used to synchronize the modifications. For example, D1, D2, and D3 are modifications to data on replica R1, and these have to be synchronized across replicas R2 and R3, D1, D2, and D3 are simultaneously synchronized with R2 and R3.
- **By Server:** Modifications to data are synchronized sequentially. Only one thread is used to sync the modifications. For example, D1, D2, and D3 are modifications to data on replica R1. These have to be synchronized across replicas R2 and R3, D1 is first synchronized with R2 and R3. Then D2 is synchronized with R2 and R3.
- **By Dynamic Adjust:** Based on the system resources you have allotted, OpenText eDirectory automatically chooses the synchronization method.

Using iMonitor, you can specify the method of synchronization using **Agent Configuration** under **Agent Synchronization**. For more information, refer to "Controlling and Configuring the DS Agent" on page 232.

NOTE: ◆Although iMonitor provides an option to specify synchronization by server method for synchronization, it is not recommended for single server tree because Skulker will not be able to synchronize the background processes.

- From eDirectory 9.2.2 and later, Skulker will not start synchronization immediately after the data transaction completes successfully. Immediate skulking may affect the performance of OpenText eDirectory operations. Therefore, synchronization of data between OpenText eDirectory servers will be delayed by 5 seconds by default. If required, you can increase the delay in synchronization by exporting the NDSD_CC_SKULK_DELAY environment variable. Values can be entered only in seconds for this variable as shown in the below example:

  ```
  NDSD_CC_SKULK_DELAY=<SECONDS>
  NDSD_CC_SKULK_DELAY=
  ```

  If the above environment variable is left blank, the synchronization will be delayed by 5 seconds by default.

Priority Sync

Priority Sync is one of the two synchronization processes in OpenText eDirectory. You can use priority sync to sync your critical data immediately without waiting for normal synchronization.

Priority sync is complimentary to the normal synchronization process in OpenText eDirectory. Unlike normal synchronization, in priority sync, the changes are not buffered before synchronizing them across the servers. This makes priority sync faster than normal synchronization.

You can sync your critical data through Priority Sync when you cannot wait for normal synchronization. The Priority Sync process is faster than the normal synchronization process. Priority Sync is supported only between two or more eDirectory 8.8 or later servers hosting the same partition.

The following table lists the platforms that support the Priority Sync feature:

Feature List	Linux	Windows
Priority Sync	✓	✓

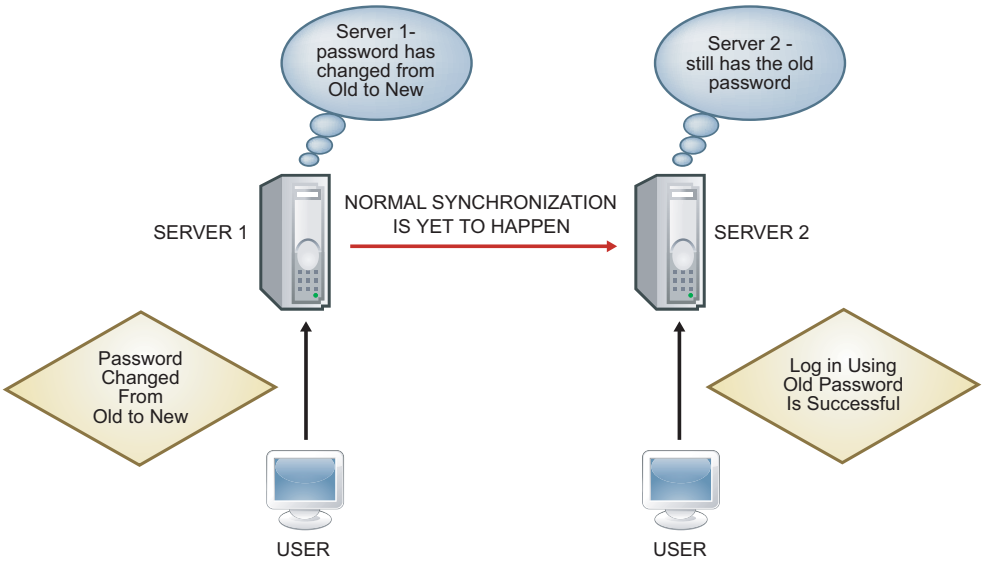
This section includes the following information:

- “Need for Priority Sync” on page 118
- “Using Priority Sync” on page 119

Need for Priority Sync

Normal synchronization can take some time, during which the modified data would not be available on other servers. For example, suppose that in your setup you have different applications talking to the directory. You change your password on Server1. With normal synchronization, it is some time before this change is synchronized with Server2. Therefore, a user would still be able to authenticate to the directory through an application talking to Server2, using the old password.

Figure 4-3 *Need for Priority Sync*



In large deployments, when the critical data of an object is modified, changes need to be synchronized immediately. The Priority Sync process resolves this issue.

Using Priority Sync

To synchronize date modifications through Priority Sync, you need to apply the Priority Sync policies to the partitions through OpenText Identity Console.

Priority sync is enabled by default. Refer to "Enabling and Disabling Inbound and Outbound Priority Sync" on page 120 for more information.

To sync the modifications to the critical data through priority sync:

1. Specify the number of threads for priority sync.

   See "Priority Sync Threads" on page 120 for more information.

2. Specify the priority sync queue size.

   See "Priority Sync Queue Size" on page 120 for more information.

3. Create and define a priority sync policy by identifying the critical attributes that you want to sync as priority.

   See "Creating and Defining a Priority Sync Policy" on page 122 for more information.

4. Apply the priority sync policy to one or more partitions.

   See "Applying a Priority Sync Policy" on page 123 for more information.

The priority sync process is to sync only the modifications to the critical attributes. Priority sync maintains the object transaction model. So, if noncritical data is modified and is not yet synchronized, and if the critical data is changed for the same entry, the noncritical data along with critical data is synchronized.

For example, a user has the following attributes: Income, Employee No, Address, and Cube No. You identify Income and Address as critical attributes. Employee No and Cube No are modified but these modifications are not yet synchronized. When the modifications to Income and Address are synchronized through priority sync, Employee No and Cube No also get synchronized, though they are not identified as critical data.

This section provides you the following information:

- "Enabling and Disabling Inbound and Outbound Priority Sync" on page 120
- "Priority Sync Threads" on page 120
- "Priority Sync Queue Size" on page 120
- "Managing Priority Sync Policies" on page 121
- "When Can Priority Sync Fail?" on page 124

Enabling and Disabling Inbound and Outbound Priority Sync

You can enable or disable the inbound and outbound priority sync in OpenText eDirectory using iMonitor. Refer to "Controlling and Configuring the DS Agent" on page 232 for more information.

Inbound priority sync is enabled by default. By disabling the inbound priority sync on a server, the modifications to the critical data on other servers are not synchronized with this server through priority sync. However, the modifications are synchronized by the normal synchronization process.

Outbound priority sync is enabled by default. By disabling this option on a server, the modifications to the critical data on this server are not synchronized with other servers through priority sync. However, the modifications are synchronized by the normal synchronization process.

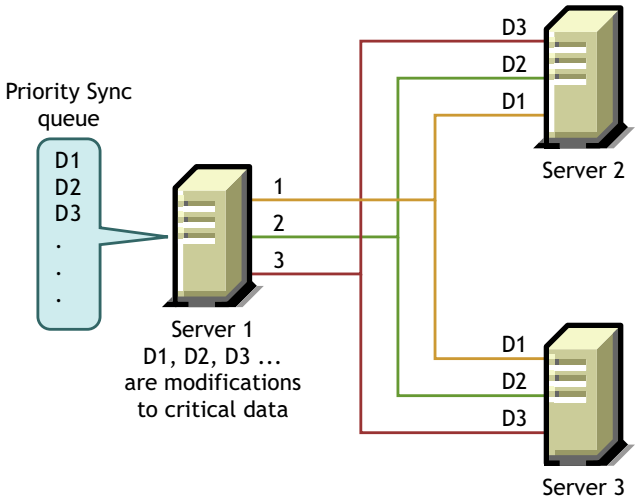
Priority Sync Threads

You need to configure the number of threads to be used for outbound priority sync. In iMonitor, you can specify the number of priority sync threads using **Agent Configuration** under **Agent Synchronization**. For more information, refer to "Controlling and Configuring the DS Agent" on page 232. The supported values are 1 to 32. The default is 4.

Priority Sync Queue Size

This indicates the maximum number of modified critical entries the queue can hold before synchronizing them. As soon as you modify the critical entries, they go into the priority sync queue and are synchronized one after the other. For example, if D1, D2, and D3 are the critical entries that are modified on server1 and these entries have to be synchronized across server2 and server3 through priority sync, then D1 is first synchronized with server2 and server3. Then D2 is synchronized with server2 and server3, and later D3 is synchronized with server2 and server3. If an earlier entry in the queue is not successfully synchronized with one of the servers, it does not affect the synchronization of the rest of the entries.

Figure 4-4 *Priority Sync Queue*



You can specify the priority sync queue size in iMonitor using **Agent Configuration** under **Agent Synchronization**. For more information, refer to "Controlling and Configuring the DS Agent" on page 232.

During a priority sync process, if a number of modifications happen at short intervals, the queue reaches its maximum size then, the queue expires and a new queue is formed. The modifications in the older queue that are not yet synchronized, will be synchronized by normal synchronization.

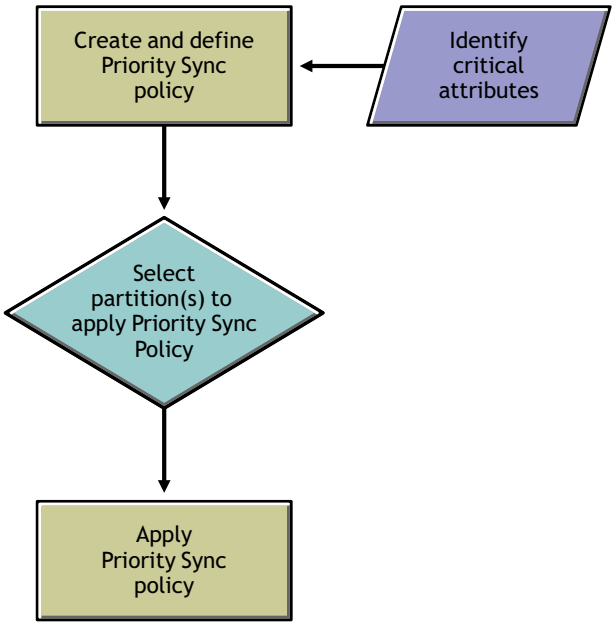
The queue size for priority sync can vary from 0 to 2^{32} - 1. By default, this value is 2^{32} - 1. If the Priority Sync queue size is set to 0, no modifications are synchronized through priority sync. These modifications are synchronized by normal synchronization.

The value -1 implies unlimited queue size. -1 is 2^{32} - 1. When a negative value is specified, for example, -3, it means -3 = -1-2, which is 2^{32} - 1-2.

Managing Priority Sync Policies

You can manage priority sync by creating and defining policies and applying them to partitions through LDAP. You define a priority sync policy by identifying the attributes that are critical.

Figure 4-5 *Priority Sync process*



For example, if the attributes Password and Account Number are critical, you can create a priority sync policy PS1 that contains these attributes. You can then apply the policy PS1 to a partition P1. If you change the password or the account number of an entry on a server, the changes are immediately synchronized with other servers having partition P1.

For priority sync to happen, you need to check if outbound and inbound priority sync are enabled in iMonitor. Inbound and outbound priority sync are enabled by default. If you disable inbound and outbound priority sync, the modifications to the data are synchronized by normal synchronization.

For more information, see “Controlling and Configuring the DS Agent” on page 232.

This section provides the following information:

- “Creating and Defining a Priority Sync Policy” on page 122
- “Editing a Priority Sync Policy” on page 123
- “Applying a Priority Sync Policy” on page 123
- “Deleting a Priority Sync Policy” on page 124

When you create a child partition, the priority sync policy that is applied to the parent is inherited by the child partition. When you merge partitions, the priority sync policy of the parent is retained.

Creating and Defining a Priority Sync Policy

You can define a priority sync policy by selecting the attributes either directly or through an object class. When you select attributes through an object class, all the attributes under the object class are selected for priority sync. You can choose to select the mandatory or optional attributes for priority sync.

The priority sync policy can be created anywhere in the OpenText eDirectory tree using LDAP.

To create an empty priority sync policy:

```
dn:cn=policy1,o=policies

changetype:add

objectclass:prsyncpolicy
```

To define the priority sync policy by marking the attributes for priority sync:

```
dn:cn=policy2,o=policies

changetype:add

objectclass:prsyncpolicy

prsyncattributes:description
```

In the above example, Description is the attribute marked for priority sync.

Editing a Priority Sync Policy

You can edit a Priority Sync Policy object using LDAP.

In the following example, the priority sync policy is modified by marking Surname for priority sync instead of Description.

```
dn:cn=policy2,o=policies

changetype:modify

add:prsyncattributes

prsyncattributes:surname
```

To remove an attribute that is marked priority sync from the priority sync policy:

```
dn:cn=policy2,o=policies

changetype:modify

add:prsyncattributes

prsyncattributes:description
```

In the above example, the attribute Description is removed from the priority sync policy.

Applying a Priority Sync Policy

You can apply one priority sync policy to many partitions, but not more than one policy to a partition.

You can apply a priority sync policy to a partition using LDAP.

To apply a priority sync policy to a root partition:

```
dn:

changetype:modify

add:prsyncpolicydn

prsyncpolicydn:cn=policy2,o=policies
```

In the above example, policy2 is applied to the root partition.

To apply a priority sync policy to a nonroot partition:

```
dn:o=org
changetype:modify
add:prsyncpolicydn
prsyncpolicydn:cn=policy2,o=policies
```

In the above example, policy2 is applied to the nonroot partition.

To replace a priority sync policy for a nonroot partition:

```
dn:o=org
changetype:modify
replace:prsyncpolicydn
prsyncpolicydn:cn=policy1,o=policies
```

In the above example, policy2 is replaced by policy1.

To disassociate a priority sync policy with a nonroot partition:

```
dn:o=org
changetype:modify
delete:prsyncpolicydn
```

In the above example, the priority sync policy is disassociated from the nonroot partition O=Org.

Deleting a Priority Sync Policy

You can delete a priority sync policy using LDAP.

```
dn:cn=policy1,o=policies
changetype:delete
```

NOTE: For more information on creating and managing priority sync policies, see "Using LDAP Tools on Linux" on page 349 and "OpenText Import Conversion Export Utility" on page 157.

When Can Priority Sync Fail?

Priority sync can fail under any of the following circumstances:

- Network failure: Priority sync will not store modifications if it is unable to send them to the remote server in the case of network failure.
- Priority sync queue size reaches its maximum: Priority sync will ignore the changes in the priority sync queue if the number of entries exceeds the priority sync queue size.
- Failure in schema synchronization: If the schema is not synchronized, priority sync process will fail.

- Object does not exist on other servers: If the creation of the object is itself not synchronized, priority sync fails.
- Mixed servers in the replica ring: If you have both eDirectory 8.8 and pre-eDirectory 8.8 servers, priority sync fails.

When priority sync fails because of any of the above reasons, the changes to the critical data are synchronized through normal synchronization.

Policy Based Replication

Replication in OpenText eDirectory follows a mesh topology, by default. This means that all replicas in a replica ring can outbound and inbound to each other. The mesh model may not be ideal in all environments. The *Policy Based Replication* allows administrators to configure the replication topology for optimizing the replication traffic.

To configure the replication topology, create a policy file and specify the policy for all the partitions in a single file and then copy it to the required servers.

On Linux

Create the policy file in XML format and name it as `selectivesync.xml` and place it along with the `nds.conf` file.

The following is a sample XML definition of a policy:

```
<?xml version="1.0" encoding="utf-8" ?>

<SelectiveSync   xmlns="http://www.novell.com/nds"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.novell.com/nds
  file:/opt/novell/eDirectory/lib64/nds-schema/xsd/selectivesync.xsd"
config-version="0.1">

    <Partition DN=".novell.TREE.">

         <SourceServer DN=".server1.novell.TREE.">

                <SyncronizeTo>.server2.novell.TREE.</SyncronizeTo>

         </SourceServer>

         <SourceServer DN=".server2.novell.TREE.">

                <SyncronizeTo>.server3.novell.TREE.</SyncronizeTo>

         </SourceServer>

         <SourceServer DN=".server3.novell.TREE.">

                <SyncronizeTo>.server1.novell.TREE.</SyncronizeTo>

         </SourceServer>

    </Partition>

</SelectiveSync>
```

On Windows

Create the policy file in XML format and name it as `selectivesync.xml` in the installed location (for example, `C:\Novell\NDS`).

The following is a sample XML definition of a policy:

```
<?xml version="1.0" encoding="utf-8" ?>

<SelectiveSync   xmlns="http://www.novell.com/nds"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.novell.com/nds
C:\Novell\NDS\selectivesync.xsd" config-version="0.1">

    <Partition DN=".novell.TREE.">

         <SourceServer DN=".server1.novell.TREE.">

                <SyncronizeTo>.server2.novell.TREE.</SyncronizeTo>

         </SourceServer>

         <SourceServer DN=".server2.novell.TREE.">

                <SyncronizeTo>.server3.novell.TREE.</SyncronizeTo>

         </SourceServer>

         <SourceServer DN=".server3.novell.TREE.">

                <SyncronizeTo>.server1.novell.TREE.</SyncronizeTo>

         </SourceServer>

    </Partition>

</SelectiveSync>
```

Note that in Windows there is no file while specifying the xsd path.

Manually Configuring Synchronization Threads

The threads created to replicate to more servers simultaneously can be increased manually by configuring the maximum number of threads created. This setting is applicable to all the partitions in a server.

To configure the maximum number of threads created:

1. Log into iMonitor.
2. Go to **Agent Configuration** > **Agent synchronization**.
3. Optionally, in the **Synchronization Method** section, select **by server**.
4. In the **System Computed Synchronization Threads** section, select **disabled**.
5. In the **Max. Manual Setting Synchronization Threads** section, set the desired number of threads.

System Computed Synchronization

In system computed synchronization, the number of skulker threads are calculated using following two formulas:

- **In partition mode**: Number of skulker threads = Number of partitions on that server
- **In server mode**: Number of skulker threads = (Number of servers known to the server + 1)/2

If **Max. System Computed Synchronization Threads** is disabled, the above two formulas will not be used. Instead the value specified for **Max. Manual Setting Synchronization Threads** will be used.

For example, consider a setup with 5 servers and 3 partitions. If you enable **Max. System Computed Synchronization Threads**: in partition mode, a server can create a maximum of 3 skulker threads and in server mode, it can create a maximum of 3 skulker threads. However, when there are a maximum of 3 skulker threads, one server cannot send updates to the other 4 servers on all partitions in parallel. In this case, disable **Max. System Computed Synchronization Threads**, and then increase the number of skulker threads in **Max. Manual Setting Synchronization Threads**.

Maximum Number of Skulker Threads

If you set **Max. Manual Setting Synchronization Threads** to 12, one server can send updates to all servers on all partitions in parallel. However, this setup cannot create more than 12 skulker threads in server mode and 3 skulker threads in partition mode even if **Max. Manual Setting Synchronization Threads** is set to a higher value than 12.

Configuring Asynchronous Outbound Synchronization

In the previous releases of OpenText eDirectory, outbound synchronization from one server to another server was performed sequentially by a single thread, which took a long time to replicate the changes.

OpenText eDirectory includes a thread that analyzes the change cache and prepares the packets to be sent across to the other server, and then fills a queue of packets. Another thread picks up the packets and sends them across to the other server one by one. This optimizes the synchronization and reduces time.

To configure outbound synchronization from one server to another server:

1. Log into iMonitor.
2. Go to **Agent Configuration** > **Background Process Settings**.
3. In the **Asynchronous Outbound Synchronization Settings** section, select **Enable**.

NOTE: Enabling asynchronous outbound synchronization may lead to increased CPU and I/O utilization at the receiving server. To avoid this, you can set a delay in sending the packets by specifying a delay interval in **Async Dispatcher Thread Delay**. You can set this delay interval between 0 to 999 milliseconds. The default value is zero milliseconds.

Configuring Background Processes

You can control the speed of the skulker, purger, and obituary background processes by using any one of the following settings:

- CPU - Specifies the maximum percentage of computer resources and the delay between two executions of the same process (skulker, purger, and obituary).
- Hard Limit - Specifies a static delay setting for the individual skulker, purger, and obituary processes.

For information about how to configure background processes, see “Configuring Background Processes” on page 235.

Hard Limit Policy

The Hard Limit Policy is enabled, by default. The background processes process a certain number of objects and then sleep for an interval of 100 milliseconds (default value). You can reduce the delay (sleep) interval to improve the performance of the system. You can increase the CPU utilization, when the delay is close to 0 milliseconds and if one, or more of these processes are running in the background. You must monitor and tune it accordingly.

CPU-Based Dynamic Policy

The CPU-based policy allows the system to dynamically tune the delay of the following three background processes to restrict the maximum CPU utilization:

- Change cache processing delay (part of outbound synchronization)
- ObitProc delay (obituary processing)
- Purger delay (pruning change cache)

The system automatically restricts CPU utilization to the configured level. When the client load is high, the background processes slow down and when the client load reduces, the speed of the background processes increase. If you don’t want the background processes to be slow, you can configure the maximum delay limit by reducing the sleep interval in this policy. However, setting a small sleep interval can cause breach of CPU restrictions.

Background Process Interval

You can set interval values for the following background processes:

- Backlink/DRL Interval
- Cleaner Interval
- Outbound Sync Interval
- Schema Sync Interval
- Janitor Interval
- Purger Interval

To configure the background process intervals:

1. Log into iMonitor.
2. Go to **Agent Configuration** > **Background Process Settings**.
3. In the **Background Process Interval** section, specify a value for interval.

5 Managing the Schema

The schema of your OpenText eDirectory tree defines the classes of objects that the tree can contain, such as Users, Groups, and Printers. It specifies the attributes (properties) that comprise each object type, including those that are required when creating the object and those that are optional.

Each OpenText eDirectory object belongs to an object class that specifies which attributes can be associated with the object. All attributes are based on a set of attribute types that are, in turn, based on a standard set of attribute syntaxes.

The OpenText eDirectory schema not only controls the structure of individual objects, but it also controls the relationship among objects in the OpenText eDirectory tree. The schema rules allow some objects to contain other subordinate objects. Thus the schema gives structure to the OpenText eDirectory tree.

You might need to make changes to your schema as your organization's informational needs change. For example, if you never required a fax number on your User object before but you need one now, you can create a new User class that has Fax Number as a mandatory attribute, then begin using the new User class to create User objects.

The Schema Management role in OpenText identity Console lets those with the Supervisor right to a tree customize the schema of that tree and perform the following tasks:

- View a list of all classes and attributes in the schema.
- Extend the schema by adding a class or an attribute to the existing schema.
- Create a class by naming it and specifying applicable attributes, flags, and containers to which it can be added, and parent classes from which it can inherit attributes.
- Create an attribute by naming it and specifying its syntax and flags.
- Add an attribute to an existing class.
- Delete a class or an attribute that is not in use or that has become obsolete.
- Identify and resolve potential problems.

This chapter contains information on the following topics:

- "Extending the Schema" on page 132
- "Viewing the Schema" on page 135
- "Manually Extending the Schema" on page 136
- "Schema Flags Added in version 8.7 Onwards" on page 139
- "Using the Client to Perform Schema Operations" on page 140

For more detailed schema information, see the *eDirectory Schema Reference* (https://www.novell.com/documentation/developer/ndslib/schm_enu/data/h4q1mn1i.html).

Extending the Schema

You can extend the schema of a tree by creating a new class or attribute. To extend the schema of your OpenText eDirectory tree, you need the Supervisor right to the entire tree.

You can extend the schema by

- Creating a Class
- Deleting a Class
- Creating an Attribute
- Adding an Optional Attribute to a Class
- Deleting an Attribute

You can extend the schema for auxiliary attributes by

- Creating an Auxiliary Class
- Extending an Object with the Properties of an Auxiliary Class
- Modifying an Object's Auxiliary Properties
- Deleting Auxiliary Properties from an Object

Creating a Class

You can add a class to your existing schema as your organizational needs change.

1. In OpenText Identity Console home page, click the **Schema Management** Tile.
2. Click **Classes** > **Create Classes** +.
3. Follow the instructions in the Create Attribute Wizard to define the object class.

   Help is available throughout the wizard. Click Help icon for more information.

   If you need to define custom properties to add to the object class, cancel the wizard and define the custom properties first. See Creating an Attribute for more information.

Deleting a Class

You can delete unused classes that aren't part of the base schema of your OpenText eDirectory tree. OpenText Identity Console only prevents you from deleting classes that are currently being used in locally replicated partitions.

You might also want to consider deleting a class from the schema in the following instances:

- After merging two trees and resolving class differences
- Any time a class has become obsolete

To delete a class:

1. In OpenText Identity Console home page, click the **Schema Management** tile.
2. Click **Classes**.
3. Select the class you want to delete.

Only the classes that are allowed to be deleted are shown.

4 Click **Delete**.

Creating an Attribute

You can define your own custom types of attributes and add them as optional attributes to existing object classes. You can't, however, add mandatory attributes to existing classes.

NOTE: Due to a replication issue, attributes in OpenText eDirectory other than the stream attribute type cannot contain values larger than 60 KB or 30,000 characters. If a user or application sets the value of a string or binary attribute and exceeds that limit, OpenText eDirectory returns a -649 error indicating that the value is too long.

To create a new attribute:

1 In OpenText Identity Console home page, click **Schema Management** tile.

2 Click **Attribute** > **Create Attribute** +

3 Follow the instructions in the Create Attribute Wizard to define the new attribute.

Help is available throughout the wizard. Click Help icon for more information.

Adding an Optional Attribute to a Class

You can add optional attributes to existing classes. This might be necessary if

- Your organization's informational needs change.
- You are preparing to merge trees.

NOTE: Mandatory attributes can only be defined while creating a class.

To add an optional attribute class:

1 On the OpenText Identity Console home page, click the **Schema Management**.

2 Click **Classes** > from the **Select Class** list, select a class.

3 On the **Class Information** page > click **Add attribute** +.

4 On the **SELECT ATTRIBUTE(S)** page, select the Attribute > click **Add**.

The Attribute is added to the Class.

5 Click **Save** > click **OK**.

If you add an attribute by mistake or change your mind, select the attribute in the **Add These Optional Attributes** list, then click to remove it from the list of attributes you want to add.

Objects you create of this class will now have the properties you added. To set values for the added properties, use the generic Other property page of the object.

TIP: You can modify an existing class by using this page to add to the **Current Attributes** list. You can remove only attributes you have added prior to clicking **OK**. You cannot remove any attribute that has been previously added and saved.

Deleting an Attribute

You can delete unused attributes that aren't part of the base schema of your OpenText eDirectory tree.

You might also want to delete an attribute from the schema in the following instances:

- After merging two trees and resolving attribute differences
- Any time an attribute has become obsolete

To delete an attribute:

1. On OpenText Identity Console home page, click the **Schema Management** tile.
2. Click **Attributes**.
3. Select the attribute you want to delete.

   Only the attributes that are allowed to be deleted are shown.
4. Click **Delete** .

Creating an Auxiliary Class

An auxiliary class is a set of properties (attributes) added to particular OpenText eDirectory object instances rather than to an entire class of objects. For example, an e-mail application could extend the schema of your OpenText eDirectory tree to include an E-Mail Properties auxiliary class and then extend individual objects with those properties as needed.

With Schema Manager, you can define your own auxiliary classes. You can then extend individual objects with the properties defined in your auxiliary classes.

To create an auxiliary class:

1. On OpenText Identity Console, click the **Roles and Tasks** button .
2. Click **Schema** > **Create Class**.
3. Specify a class name and (optional) ASN1 ID, then click **Next**.
4. Select **Auxiliary Class** when setting the class flags, then click **Next**.
5. Follow the instructions in the Create Class Wizard to define the new auxiliary class.

   Help is available throughout the wizard.

Extending an Object with the Properties of an Auxiliary Class

1. On OpenText Identity Console home page, click the **Schema Management** tile.
2. Click **Object Extensions** > **Object Name** .
3. Select a user or group, then click **Current aux class extensions** .

   Object extended successfully message appears.
4. Click **OK**.
5. Depending on whether the auxiliary class that you want to use is already listed under **Current Auxiliary Class Extensions**, complete the appropriate action:

Auxiliary Class Already Listed?	Action
Yes	Quit this procedure. See "Modifying an Object's Auxiliary Properties" on page 135 instead.
No	Click **Add**, select the auxiliary class, then click **OK**.

6 Click **Close**.

Modifying an Object's Auxiliary Properties

1 On OpenText Identity Console home page, click the **Object Management** tile.

2 On the **Objects** list, select the Object > **Modify Object**.

3 Specify the name and context of the object you want to modify, then click **Save**.

4 On the screen that appears, set the attribute values you want.

- Click any unvalued attributes to add them to the list of valued attributes.
- Select a valued attribute, then click **Modify** to edit the attribute, or **Delete** to remove the attribute.
- You must know the syntax of a property to set it correctly. For more information, see the *OpenText eDirectory Schema Reference* (http://developer.novell.com/documentation/ndslib/schm_enu/data/h4q1mn1i.html).

5 Click **Save**, then click **OK**.

Deleting Auxiliary Properties from an Object

1 On OpenText Identity Console home page, click the **Schema Management** tile

2 Click **Object Extensions**.

3 In the list of current auxiliary class extensions select the **Current aux class extension** whose properties you want to delete, then click **Delete**.

Axillary class deleted successfully.

4 Click **OK**.

This deletes all the properties added by the auxiliary class except for any that the object already had innately.

5 Click **Close**.

Viewing the Schema

You can view the schema to evaluate how well the schema meets your organization's informational needs. The larger and more complex your organization, the more likely it is that you need to customize the schema, but even small organizations might have unique tracking needs. Viewing the schema can help you determine what, if any, extensions you need to make to the base schema.

Viewing Class Information

The Class Information page in OpenText Identity Console displays information about the selected class and lets you add attributes. Most of the information displayed on the page was specified when the class was created. Some of the optional attributes might have been added later.

During class creation, if the class was specified to inherit attributes from another class, the inherited attributes are classified as they are in the parent class. For instance, if Object Class is a mandatory attribute for the parent class, then it displays on this screen as a mandatory attribute for the selected class.

1. On OpenText Identity Console home page, click the **Schema Management** tile.
2. Click **Classes** > select the **Class** you want information on.

   Click for more information.

Viewing Attribute Information

1. On OpenText Identity Console home page, click the **Schema Management** tile.
2. Click **Attributes** > select the **Attribute** you want information on.

   Click for more information.

Manually Extending the Schema

You can manually extend the OpenText eDirectory schema using files with a `.sch` extension.

This section contains the following information:

- "Extending the Schema on Windows" on page 136
- "Extending the Schema on Linux" on page 137

Extending the Schema on Windows

Use `NDSCons.exe` to extend the schema on Windows servers. Schema files (`*.sch`) that come with OpenText eDirectory are installed by default into the `C:\Novell\NDS` directory.

1. Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**.
2. Click **install.dlm**, then click **Start**.
3. Click **Install Additional Schema Files**, then click **Next**.
4. Log in as a user with administrative rights, then click **OK**.
5. Specify the schema file path and name.
6. Click **Finish**.

Extending the Schema on Linux

The following sections provide information about extending the schema on Linux computers:

- "Using the ndssch Utility to Extend the Schema on Linux" on page 137
- "Extending the RFC 2307 Schema" on page 137

Using the ndssch Utility to Extend the Schema on Linux

In addition to OpenText Identity Console, you can use ndssch, the OpenText eDirectory schema extension utility, to extend the schema on Linux computers. The attributes and classes that you specify in the schema file (`.sch`) will be used to modify the schema of the tree. The association between the attributes and classes are created as specified in the `.sch` file.

Use the following syntax:

```
ndssch [-h hostname[:port]] [-t tree_name] [-F <logfile>] admin-FDN
schemafile...
```

```
ndssch [-h hostname[:port]] [-t tree_name] [-d] admin_FDN schemafile
[schema_description]...
```

ndssch Parameter	Description
`-h hostname`	Name or IP address of the server that the schema is to be extended on. The schema of the tree that the specified server belongs to will be extended. This is an optional parameter if the tree is located on the host whose schema is to be extended. Otherwise, it is a mandatory parameter.
`port`	The server port.
`-t tree_name`	Name of the tree that the schema is to be extended on. This is an optional parameter. The default tree name is the one specified in the `/etc/opt/novell/eDirectory/conf/nds.conf` file. For more information, see "Configuration Parameters" in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.
`-F logfile`	Specifies the path name to the ndssch log file.
`admin-FDN`	Name with the full context of the user with OpenText eDirectory administrator rights to the tree.
`schemafile`	Filename that contains information about the schema to be extended.
`-d, schema_description`	When this option is used, every schema file must be followed by a description of the schema file.

Extending the RFC 2307 Schema

The attributes and object classes defined in RFC 2307 (http://www.ietf.org/rfc/rfc2307.txt) are user or group related and NIS related. The user- or group-related definitions are compiled into the `/opt/novell/eDirectory/lib/nds-modules/schema/rfc2307-usergroup.sch` file. The NIS-related definitions are compiled into the `/opt/novell/eDirectory/lib/nds-modules/`

schema/rfc2307-nis.sch file. The corresponding files in the LDIF format are also provided (/opt/novell/eDirectory/lib/nds-modules/schema/rfc2307-usergroup.ldif and /opt/novell/eDirectory/lib/nds-modules/schema/rfc2307-nis.ldif respectively).

You can extend the RFC 2307 schema using the ndssch utility or the ldapmodify tool.

- "Using the ndssch Utility" on page 138
- "Using the ldapmodify Utility" on page 138

Using the ndssch Utility

Enter one of the following commands:

```
ndssch -t tree_name admin-FDN /opt/novell/eDirectory/lib/nds-schema/
rfc2307-usergroup.sch
```

or

```
ndssch -t tree_name admin-FDN /opt/novell/eDirectory/lib/nds-schema/
rfc2307-nis.sch
```

Parameter	Description
-t	Name of the tree on that the schema is to be extended on. This is an optional parameter. If this parameter is not specified, the tree name is taken from the /etc/opt/novell/eDirectory/conf/nds.conf file.

Using the ldapmodify Utility

Enter one of the following commands:

```
ldapmodify -h -D -w -f /opt/novell/eDirectory/lib/nds-schema/rfc2307-
usergroup.ldif
```

or

```
ldapmodify -h -D -w -f /opt/novell/eDirectory/lib/nds-schema/rfc2307-
nis.ldif
```

Parameter	Description
-h *ldaphost*	Specifies an alternate host on which the LDAP server is running.
-D *binddn*	Uses *binddn* to bind to the X.500 directory. It should be a string-represented DN as defined in RFC 1779.
-w *passwd*	Uses *passwd* as the password for simple authentication.
-f *file*	Reads the entry modification information from file instead of from standard input.

Schema Flags Added in version 8.7 Onwards

The READ_FILTERED and BOTH_MANAGED schema flags were added to version 8.7 and above.

READ_FILTERED is used to indicate that an attribute is an LDAP OPERATIONAL attribute. LDAP uses this flag when it requests to read the schema to indicate that an attribute is “operational.” Some internally defined schema attributes now have this flag set. The LDAP “operational” definition includes three schema flags. In addition to the new READ_FILTERED flag, the other existing flags that are used to indicate “operational” are the READ_ONLY flag and the HIDDEN flag. If any of these flags is present on a schema definition, LDAP treats the attribute as “operational” and will not return that attribute unless specifically requested to do so.

BOTH_MANAGED is a new security rights enforcement mechanism. It is only meaningful on an attribute of Distinguished Name syntax. If set on such an attribute, it will require that the requesting connection have rights on both the target object and attribute and the object being referenced by the target attribute. This is an expansion of the current WRITE_MANAGED flag functionality. This flag is not currently set on any base schema attributes. This new security behavior will only occur on an OpenText eDirectory 8.7.*x* server or later versions, so for consistent behavior relating to this flag, the entire tree must be upgraded to 8.7 or later versions of the product.

Because only 8.7.*x* (or later versions) server will recognize these new flags, they can be set only on a schema definition by the version 8.7.*x* (or later versions) server which holds a copy of the root partition (because only servers holding root can do schema modifications). The normal installation of a new server or upgrading an existing server that doesn’t hold the root partition will not successfully add these new flags to the schema in your tree.

If you want either of these new features enabled in your tree, you need to ensure that the schema is successfully extended to add these new flags. There are two ways to do this. The first option is to choose a server that holds a writable copy of the root partition to be upgraded to version 8.7 or later. This will automatically extend the schema correctly with the new flags.

The second option is more involved and contains the following steps:

1. Install a new 8.7.*x* (or later version) server or upgrade any existing server in the tree. This server does not need to hold a copy of [Root].
2. Manually add a copy of the root partition to this new server.
3. Rerun the appropriate schema extension files on that server to extend the schema:

Platform	Instructions
Windows	Load `install.dlm`, then click **Install Additional Schema Files**.
Linux	Use the ndssch utility. See “Using the ndssch Utility to Extend the Schema on Linux” on page 137 for more information.

4. Install the new schema files you choose that have these new flags set.
5. (Optional) After the schema has synchronized, you can remove the root replica from this server.

NOTE: These new schema flags enable optional features. If you don’t need or want the new functionality, the absence of these new flags on the schema definitions will not cause any problems in the normal operation of OpenText eDirectory in your tree. In the case of the READ_FILTERED flag, it would not be present on some attribute definitions. Therefore, an LDAP read request for all

attributes of an object might get some extra data it would not otherwise have received. Some attributes will still be treated as operational anyway because of the presence of the READ_ONLY or HIDDEN flag. The BOTH_MANAGED flag is intended only to be enabled on fully upgraded trees, because consistent operation of this feature can be achieved only in that environment.

Using the Client to Perform Schema Operations

The OpenText eDirectory Management Toolbox (eMBox) Client is a command line Java client that gives you remote access to DSSchema operations. You can use the DSSchema eMTool to synchronize schema, import remote schema, declare a new schema epoch, reset the local schema, and perform a global schema update (operations normally performed using DSRepair. For more information, see "Maintaining the Schema" on page 322.).

The `emboxclient.jar` file is installed on your server as part of OpenText eDirectory. You can run it on any machine with a JVM. For more information on the Client, see "Using the Command Line Client" on page 560.

Using the DSSchema eMTool

1 Run the Client in interactive mode by entering the following at the command line:

```
java -cp path_to_the_file/emboxclient.jar  -i
```

(If you have already put the `emboxclient.jar` file in your class path, you only need to enter `java -i`.)

The Client prompt appears:

```
Client>
```

2 Log in to the server you want to repair by entering the following:

```
login -sserver_name_or_IP_address -pport_number
-uusername.context -wpassword -n
```

The port number is usually 80 or 8028, unless you have a Web server that is already using the port. The `-n` option opens a nonsecure connection.

The Client indicates whether the login is successful.

3 Enter a repair command, using the following syntax:

```
dsschema.task options
```

For example:

`dsschema.rst` requests the master replica of the root of the tree to synchronize its schema to this server.

`dsschema.irs -n MyTree` imports remote schema from the tree named MyTree.

A space must be between each switch. The order of the switches is not important.

The Client will indicate whether the repair is successful.

See "DSSchema eMTool Options" on page 141 for more information on the DSSchema eMTool options.

4 Log out from the Client by entering the following command:

```
logout
```

5 Exit the Client by entering the following command:

```
exit
```

DSSchema eMTool Options

The following tables lists the DSSchema eMTool options. You can also use the `list -t dsschema` command in the Client to list the DSSchema options with details. See “Listing eMTools and Their Services” on page 563 for more information.

Option	Description
`rst`	Synchronizes the schema of the master replica of the root of the tree to this server.
`irs -n`*`tree_name`*	Imports remote schema from another tree.
`dse`	Declares a new schema epoch on the server that holds the master replica of root.
`rls`	Resets the local schema with a copy from the server with the master replica of the root partition.
`gsu`	Performs a global schema update.
`scc`	Adds schema circular containment rules for the Domain class.

6 Managing Partitions and Replicas

Partitions are logical divisions of the OpenText eDirectory database that form a distinct unit of data in the OpenText eDirectory tree for administrators to store and replicate OpenText eDirectory information. Each partition consists of a container object, all objects contained in it, and the information about those objects. Partitions do not include any information about the file system or the directories and files contained there.

Instead of storing a copy of the entire OpenText eDirectory database on each server, you can make a copy of the OpenText eDirectory partition and store it on many servers across the network. Each copy of the partition is known as a replica. You can create any number of replicas for each OpenText eDirectory partition and store them on any server. The types of replicas include master, read/write, read-only, subordinate references, filtered read/write, and filtered read-only.

The following table describes the replica types.

Replica	Description
Master, read/write, and read-only	Contain all objects and attributes for a particular partition.
Subordinate references	Used for tree connectivity.
Filtered replicas	Contains a subset of information from the entire partition, consisting of only the desired classes and attributes—which are defined by the server's replication filter. This filter is used to identify the classes and attributes allowed to pass during inbound synchronization and local changes. Filtered replicas allow administrators to create sparse and fractional replicas. ◆ Sparse replicas contain only the object classes that you specify. ◆ Fractional replicas contain only the attributes you specify. The functionality of filtered replicas enables fast response when the data stored in OpenText eDirectory is procured by applications. Filtered replicas also allow more replicas to be stored on a single server.
Read/write filtered replicas	Allows local modifications to classes and attributes that are a subset of the server's replication filter. However, these replicas can create objects only if all mandatory attributes for the class are within the replication filter.
Read-only filtered replicas	Does not allow local modifications.

This chapter describes how to manage partitions and replicas.

- “Creating a Partition” on page 144
- “Merging a Partition” on page 144

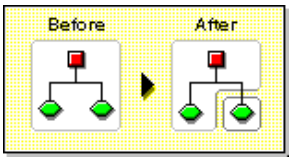
- “Moving Partitions” on page 146
- “Canceling Create or Merge Partition Operations” on page 147
- “Administering Replicas” on page 147
- “Setting Up and Managing Filtered Replicas” on page 150
- “Viewing Partitions and Replicas” on page 153

Creating a Partition

When you create partitions, you make logical divisions of your tree. These divisions can be replicated and distributed among different OpenText eDirectory servers in your network.

When you create a new partition, you split the parent partition and end up with two partitions. The new partition becomes a child partition, as seen in the following illustration.

Figure 6-1 *Before and After a Partition Split*



For example, if you choose an Organizational Unit and create it as a new partition, you split the Organizational Unit and all of its subordinate objects from its parent partition.

The Organizational Unit you choose becomes the root of a new partition. The replicas of the new partition exist on the same servers as the replicas of the parent, and objects in the new partition belong to the new partition’s root object.

Creating a partition might take some time, because all of the replicas need to be synchronized with the new partition information. If you attempt another partition operation while a partition is still being created, you receive a message telling you that the partition is busy.

You can look at the replica list for the new partition and know that the operation is complete when all replicas in the list are in an On state. You must manually refresh the view periodically because the states are not automatically refreshed.

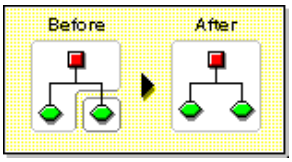
To create a partition:

1. On the OpenText Identity Console home page, click **Partition Management** tile.
2. Click **Create Partition** + > select the **Container** .
3. Specify the name and context of the container you want to create a new partition from, then click **Create**.

Merging a Partition

When you merge a partition with its parent partition, the chosen partition and its replicas combine with the parent partition. You do not delete partitions — you only merge and create partitions to define how the directory tree is split into logical divisions, as shown in the following illustration.

Figure 6-2 *Before and After a Partition Merge*



There are several reasons you might want to merge a partition with its parent:

- The directory information in the two partitions is closely related.
- You want to delete a subordinate partition, but you don't want to delete the objects in it.
- You're going to delete the objects in the partition.
- You want to delete all replicas of the partition. Merging a partition with its parent is the only way to delete the partition's master replica.
- After moving a container, which must be a partition root with no subordinate partitions, you don't want the container to be a partition anymore.
- You experience changes in your company organization, so you want to redesign your directory tree by changing the partition structure.

Consider keeping partitions separate if the partitions are large and contain hundreds of objects, because large partitions slow down network response time.

The root-most partition in the tree cannot be merged because it is the top partition and has no parent partition to merge with.

The partition is merged when the process is completed on the servers. The operation could take some time to complete depending on partition sizes, network traffic, server configuration, etc.

IMPORTANT: Before merging a partition, check the partition synchronization of both partitions and fix any errors before proceeding. By fixing the errors, you can isolate problems in the directory and avoid propagating the errors or creating new ones.

Make sure all servers that have replicas (including subordinate references) of the partition you want to merge are up before attempting to merge a partition. If a server is down, OpenText eDirectory won't be able to read the server's replicas and won't be able to complete the operation.

If you receive errors in the process of merging a partition, resolve the errors as they appear. Don't try to fix the error by continuing to perform operations—doing so only results in more errors.

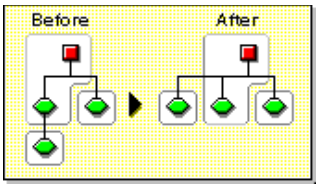
To merge a child partition with its parent partition:

1. On the OpenText Identity Console home page, click **Partition Management** tile.
2. On the **Partitions** page > select the **Partition** > **Merge Partition** icon.
3. Specify the name and context of the partition you want to merge with its parent partition, then click **OK**.

Moving Partitions

Moving a partition lets you move a subtree in your directory tree. You can move a partition root object (which is a container object) only if it has no subordinate partitions.

Figure 6-3 *Before and After a Partition Move*



When you move a partition, you must follow OpenText eDirectory containment rules. For example, you cannot move an Organizational Unit directly under the root of the current tree, because the root's containment rules allow Locality, Country, or Organization, but not Organizational Unit.

When you move a partition, OpenText eDirectory changes all references to the partition root object. Although the object's common name remains unchanged, the complete name of the container (and of all its subordinates) changes.

When you move a partition, you should choose the option to create an Alias object in place of the container you're moving. Doing so allows users to continue to log in to the network and find objects in the original directory location.

The Alias object that is created has the same common name as the moved container and references the new complete name of the moved container.

IMPORTANT: If you move a partition and do not create an Alias object in place of the moved partition, users who are unaware of the partition's new location cannot easily find that partition's objects in the directory tree, because they look for them in their original directory location.

This might also cause client workstations to fail at login if the workstation `NAME CONTEXT` parameter is set to the original location of the container in the directory tree.

Because the context of an object changes when you move it, users whose name context references the moved object need to update their `NAME CONTEXT` parameter so that it references the object's new name.

To automatically update users' `NAME CONTEXT` after moving a container object, use the NCUPDATE utility.

After moving the partition, if you don't want the partition to remain a partition, merge it with its parent partition.

Make sure your directory tree is synchronizing correctly before you move a partition. If you have any errors in synchronization in either the partition you want to move or the destination partition, do not perform a move partition operation. First, fix the synchronization errors.

To move a partition:

1. On the OpenText Identity Console home page, click **Partition Management** tile.
2. On the **Partitions** page > select the **Partition** > **Move Partition** icon.

3 Specify the name and context of the partition object you want to move in the **Object Name** field.

4 Specify the container name and context you want to move the partition to in the **Move To** field.

5 If you want to create an Alias in the old location for the partition being moved, select **Create an Alias in Place of Moved Object**.

This allows any operations that are dependent on the old location to continue uninterrupted until you can update those operations to reflect the new location.

6 Click **OK**.

Canceling Create or Merge Partition Operations

You can cancel a Create or Merge partition operation if the operation has not yet progressed past the stage at which the change is committed. Use this feature to back out of an operation, or if your OpenText eDirectory network returns OpenText eDirectory errors or fails to synchronize following a partition operation.

If replicas in your directory tree experience synchronization errors, an abort operation might not always solve the problem. However, you can use this feature as an initial troubleshooting option.

If a partition operation cannot be completed because a server is down (or otherwise unavailable), either make the server visible to the network so the operation can complete or attempt to abort the operation. If OpenText eDirectory cannot synchronize because the database is corrupted, you should abort any partition operation in progress.

Partition operations can take considerable time to fully synchronize across the network, depending on the number of replicas involved, the visibility of servers involved, and the existing wire traffic.

If you get an error that says a partition is busy, it doesn't mean that you should abort the operation. You can usually expect partition operations to complete within 24 hours depending on the size of the partition, connectivity issues, etc. If a particular operation fails to complete within this time frame, you should then attempt to abort the operation in progress.

Administering Replicas

Before you add or delete a replica, or change replica type, carefully plan target replica locations. See "Guidelines for Replicating Your Tree" on page 84.

Adding a Replica

Add a replica to a server to provide your directory with

- Fault tolerance
- Faster access to data
- Faster access across a WAN link
- Access to objects in a set context (using bindery services)

To add a replica:

1. On the OpenText Identity Console home page, click **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the server > On the **Replica View** page, click **Add Replica** + > specify the **Partition Name** and server you want to replicate.
4. Choose one of the following replica types:

Replica Type	Description
Read-Write	Users will be able to both read and modify the contents of the new replica. Select this option if there are no modifiable replicas close enough to the users who manage the OpenText eDirectory objects in this partition.
Read-Only	Users will be able to read but not modify the contents of the new replica. Select this option if there are no replicas close enough to the users who read but don't modify the OpenText eDirectory objects in this partition.
Filtered Read-Write	Users will be able to both read and modify the contents of the new replica, and the contents will be limited to the types of OpenText eDirectory objects and properties specified in a filter.
Filtered Read-Only	Users will be able to read but not modify the contents of the new replica, and the contents will be limited to the types of OpenText eDirectory objects and properties specified in a filter.

5. Click **OK**.

For more information, see "Replica Types" on page 57.

Deleting a Replica

Deleting a replica removes the replica of the partition from a server.

If you want to remove a server from the directory tree, you can delete replicas from the server before removing the server. Deleting the replicas reduces the chance of having problems when removing the server.

You can also reduce synchronization traffic on the network by removing replicas. Keep in mind that you probably don't want more than six replicas of any partition.

You cannot delete a master replica or a subordinate reference.

If the replica you want to delete is a master, you have two options:

- Go to a server with another replica of the partition and make it the new master replica

  This automatically changes the original master replica to a read/write replica, which you can then delete.

- Merge the partition with its parent partition

  This merges the replicas of the partition with those of its parent and removes them from the servers they reside on. Merging removes partition boundaries, but not the objects. The objects continue to exist on each server which held a replica of the “joined” partition.

When you delete replicas, keep the following guidelines in mind:

- For fault tolerance, you should maintain at least three replicas of each partition on different servers.
- Deleting a replica deletes a copy of part of the directory database on the targeted server.

  The database can still be accessed on other servers in the network, and the server that the replica was on still functions in OpenText eDirectory.

  You cannot delete or manage subordinate reference replicas. They are created automatically on a server by OpenText eDirectory when the server contains a replica of a partition but not of that partition’s child.

To delete a replica:

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search
3. Select the partition or server that holds the replica you want to delete.
4. Click .

Changing a Replica Type

Change a replica type to control access to the replica information. For example, you might want to change an existing read/write replica to a read-only replica to prevent users from writing to the replica and modifying directory data.

You can change the type of a read/write or a read-only replica. You cannot change the type of a master replica, but a read/write or read-only can be changed to a master, which automatically changes the original master to a read/write replica.

Most replicas should be read/write. Read/write replicas can be written to by client operations. They send out information for synchronization when a change is made. Read-only replicas cannot be written to by client operations. However, they are updated when the replicas synchronize.

You cannot change the replica type of a subordinate reference. To place a replica of a partition on a server which currently has a subordinate reference requires an Add replica operation. A subordinate reference replica is not a complete copy of a partition. The placement and management of subordinate reference replicas is handled by OpenText eDirectory. They are created automatically on a server by OpenText eDirectory when the server contains a replica of a partition but not of that partition’s child.

To change a replica type:

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the partition or server that holds the replica you want to change.
4. On the **Replica View** page > select the replica type you want to change.

5 Change the replica type, then click **OK**.

Replica Type	Description
Master	Users can both read and modify the contents of this replica, and the replica is the starting point for any future partitioning activity that affects this partition, such as creating or merging a subpartition. Only one master replica is allowed per partition.
Read-Write	Users can both read and modify the contents of the new replica. Select this option if there are no modifiable replicas close enough to the users who manage the OpenText eDirectory objects in this partition.
Read-Only	Users can read but not modify the contents of the new replica. Select this option if there are no replicas close enough to the users who read but don't modify the OpenText eDirectory objects in this partition.
Filtered Read-Write	Users can both read and modify the contents of the new replica, and the contents are limited to the types of OpenText eDirectory objects and properties specified in a filter.
Filtered Read-Only	Users can read but not modify the contents of the new replica, and the contents are limited to the types of OpenText eDirectory objects and properties specified in a filter.

The **Replica type changed successfully!** message appears.

6 Click **OK**.

For more information, see "Replica Types" on page 57.

Setting Up and Managing Filtered Replicas

Filtered replicas maintain a filtered subset of information from an OpenText eDirectory partition (objects or object classes along with a filtered set of attributes and values for those objects).

Administrators generally use the filtered replica capability to create an OpenText eDirectory server that holds a set of filtered replicas that contain only specific objects and attributes that they want synchronized.

To do this, OpenText Identity Console provides tools to create a filtered replica partition scope and filter. A scope is simply the set of partitions where you want replicas placed on a server. A replication filter contains the set of OpenText eDirectory classes and attributes you want to host on a server's set of filtered replicas. The result is an OpenText eDirectory server that can house a well-defined data set from many partitions in the tree.

The descriptions of the server's partition scope and replication filters are stored in OpenText eDirectory, and they can be managed through the Server object or the Partition and Replicas role in OpenText Identity Console.

- "Using the Filtered Replica Wizard" on page 151
- "Defining a Partition Scope" on page 151
- "Setting Up a Server Filter" on page 152

Using the Filtered Replica Wizard

The Filtered Replica Wizard guides you step-by-step through the setup of a server's replication filter and partition scope.

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the server that you want to configure a filtered replica.
4. On the **Replica View** page, click **Add Replica** + > click  to select the partition container.

   The replication filter contains the set of OpenText eDirectory classes and attributes you want to host on this server's set of filtered replicas. For more information on defining a filter set, see "Setting Up a Server Filter" on page 152.
5. Change the replica type. Example: Filtered read-write, Filtered read-only, and so on.
6. Click **OK**.
7. To define the partition scope for this server, click **Edit**.

   For more information on partition scopes, see "Defining a Partition Scope" on page 151.
8. Click **OK**.

Defining a Partition Scope

A partition scope is the set of partitions where you want replicas placed on a server. The Replica View page in OpenText Identity Console provides a view of the hierarchy of partitions in the OpenText eDirectory tree. You can select individual partitions, a set of partitions of a given branch, or all of the partitions in the tree. You can then select the type of replicas of these partitions you want added to the server, or change existing replica types.

A server can hold both full replicas and filtered replicas. For more information, see "Filtered Replicas" on page 60.

Viewing Replicas on an OpenText eDirectory Server

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.

   On the **Partitions** page you can view the list of replicas for this server.

Adding a Filtered Replica to an OpenText eDirectory Server

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the server you want to add a filtered replica to.
4. On the **Replica View** page > click **Add Replica**.
5. Specify the partition name and context.
6. Click **Filtered Read-Write** or **Filtered Read-Only,** then click **OK**.

Changing a Full Replica into a Filtered Replica

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the partition or server that holds the replica you want to change, then click **OK**.
4. Click the replica type (in the **Type** column) of the replica you want to change.
5. Click **Filtered Read-Write** or **Filtered Read-Only**, then click **OK**.

Setting Up a Server Filter

A server replication filter contains the set of OpenText eDirectory classes and attributes you want to host on a server's set of filtered replicas. You can set up a filter from any Server object. For filtered replicas, you can have only one filter per server. This means that any filter defined for an OpenText eDirectory server applies to all filtered replicas on that server. The filter, however, does not apply to full replicas.

A server's filter man be modified if required, but the operation generates a resynchronization of the replica and can thus be time consuming. Careful planning of the server's function is recommended.

You can set up or modify a server filter in either of the following ways:

- “Using the Replica View” on page 152
- “Using the Server Object” on page 152

Using the Replica View

1. On the OpenText Identity Console home page click the **Partition Management** tile.
2. On the **Partitions** page > from **Type** drop down list > select Server > Click Search.
3. Select the name and context of the partition or server that holds the replica you want to change, then click **OK**.
4. Click Edit in the Filter column for the server or partition you want to modify.
5. Add the classes and attributes you want, then click **OK**.

Using the Server Object

1. On the OpenText Identity Console home page click the **Object Management** tile.
2. Click **Directory Administration** > **Modify Object**.
3. Specify the name and context of the server that holds the replica you want to change, then click **OK**.
4. Click the **Replica** tab.
5. If no filter has been defined for this server, click **The Filter is Empty** to open the Edit Filter Dialog window, then add the classes and attributes you want.

   or

Click **Copy Filter From** to browse for an object (such as another server) whose filter you want to copy.

6 To edit an existing filter, click any hyperlinked item in the filter to open the Edit Filter Dialog window, then add or remove the classes and attributes you want.

Viewing Partitions and Replicas

This section contains the following information:

- "Viewing the Partitions on a Server" on page 153
- "Viewing a Partition's Replicas" on page 153
- "Viewing Information about a Partition" on page 154
- "Viewing Partition Hierarchy" on page 154
- "Viewing Information about a Replica" on page 154

Viewing the Partitions on a Server

You can use OpenText Identity Console to view which partitions are allocated to a server. You might want to view the partitions stored on a server if you are planning to remove a Server object from the directory tree. In this case, you can view the replicas you need to remove before removing the object.

1 On the OpenText Identity Console home page, click the **Partition Management** tile.

2 Select the name and context of a Server object.

3 On the **Partitions** page view the partitions.

Viewing a Partition's Replicas

This operation lets you identify

- Which servers the partition's replicas reside on
- Which server hosts the master replica of the partition
- Which servers have read/write, read-only, and subordinate reference replicas of the partition
- The state of each of the partition's replicas

To view a partition's replicas:

1 On the OpenText Identity Console home page, click the **Partition Management** tile.

2 On the **Partitions** page > Select the partition.

View the information on Replica View page.

Viewing Information about a Partition

The most significant reason to view information about a partition is to see its synchronization information (last successful synchronization and last attempted synchronization).

1. On the OpenText Identity Console home page, click the **Partition Management** tile.
2. On the **Partitions** page > Select the partition.
3. On the **Replica View** page > select the server.

   The **Replica information** page appears.

Viewing Partition Hierarchy

You can view the partition hierarchy in OpenText Identity Console. On the home page click **Tree View** tile to view which partitions are parent, and which are child partitions.

Viewing Information about a Replica

The most significant reason to view information about a replica is to see its state. An OpenText eDirectory replica can be in various states depending on the partition or replication operations it is undergoing. The following table describes the replica states that you might see in OpenText Identity Console.

State	Means That the Replica Is
On	Currently not undergoing any partition or replication operations
New	Being added as a new replica on the server
Dying	Being deleted from the server
Dead	Done being deleted from the server
Master Start	Being changed to a master replica
Master Done	Done being changed to a master replica
Change Type	Being changed to a different type of replica
Locked	Locked in preparation for a partition move or repair operation
Transition Move	Starting into a partition move operation
Move	In the midst of a partition move operation
Transition Split	Starting into a partition split operation (creation of a child partition)
Split	In the midst of a partition split operation (creation of a child partition)
Join	Being merged into its parent partition
Transition On	About to return to an On state
Unknown	In a state not known to OpenText Identity Console

To view information about a replica:

1. On the OpenText Identity Console home page, click the **Partition Management** tile.
2. On the **Type** drop-down menu > select the server> click **Search**.
3. Click on the required Replica to view information about that replica.

7 OpenText eDirectory Management Utilities

This chapter contains information on the following OpenText eDirectory utilities:

- "OpenText Import Conversion Export Utility" on page 157
- "Index Manager" on page 187
- "OpenText eDirectory Service Manager" on page 189
- "Offline Bulkload Utility" on page 191
- "LDIF Files" on page 200

OpenText Import Conversion Export Utility

The OpenText Import Conversion Export utility lets you

- Import data from LDIF files to an LDAP directory.
- Export data from the LDAP directory to an LDIF file.
- Migrate data between LDAP servers.
- Perform a schema compare and update.
- Load information into OpenText eDirectory using a template.
- Import schema from SCH files to an LDAP directory.

The OpenText Import Conversion Export utility manages a collection of handlers that read or write data in a variety of formats. Source handlers read data, and destination handlers write data. A single executable module can be both a source and a destination handler. The engine receives data from a source handler, processes the data, then passes the data to a destination handler.

For example, if you want to import LDIF data into an LDAP directory, the OpenText Import Conversion Export engine uses an LDIF source handler to read an LDIF file and an LDAP destination handler to send the data to the LDAP directory server. For more information on LDIF file syntax, structure, and debugging, see Appendix H, "Troubleshooting," on page 811.

You can run the OpenText Import Conversion Export client utility from the command line or from the Import Convert Export Wizard in OpenText Identity Console. The comma-delimited data handler, however, is available only in the command line utility and OpenText Identity Console.

You can use the OpenText Import Conversion Export utility in any of the following ways:

- "Using the Command Line Interface" on page 158

Both the wizard and the command line interface give you access to the OpenText Import Conversion Export engine, but the command line interface gives you greater options for combining source and destination handlers.

The OpenText Import Conversion Export utility replaces both the BULKLOAD and ZONEIMPORT utilities included with previous versions of NDS and OpenText eDirectory.

Using the Command Line Interface

You can use the command line version of the OpenText Import Conversion Export utility to perform the following:

- LDIF imports
- LDIF exports
- Comma-delimited data imports
- Comma-delimited data exports
- Data migration between LDAP servers
- Schema compare and update
- Load information into OpenText eDirectory using a template
- Schema imports

A Windows version (`ice.exe`) is included in the installation. On Linux computers, the Import/Export utility is included in the `NOVLice` package.

OpenText Import Conversion Export Syntax

The OpenText Import Conversion Export utility is launched with the following syntax:

```
ice general_options
-S[LDIF | LDAP | DELIM | LOAD | SCH] source_options
-D[LDIF | LDAP | DELIM] destination_options
```

or when using the schema cache:

```
ice -C schema_options
-S[LDIF | LDAP] source_options
-D[LDIF | LDAP] destination_options
```

When performing an update using the schema cache, an LDIF file is not a valid destination.

General options are optional and must come before any source or destination options. The `-S` (source) and `-D` (destination) handler sections can be placed in any order.

The following is a list of the available source and destination handlers:

- "LDIF Source Handler Options" on page 160
- "LDIF Destination Handler Options" on page 161
- "LDAP Source Handler Options" on page 162
- "LDAP Destination Handler Options" on page 164
- "DELIM Source Handler Options" on page 165
- "DELIM Destination Handler Options" on page 167
- "SCH Source Handler Options" on page 167
- "LOAD Source Handler Options" on page 168

General Options

General options affect the overall processing of the OpenText Import Conversion Export engine.

Option	Description
`-C`	Specifies that you are using the schema cache to perform schema compare and update.
`-l log_file`	Specifies a filename where output messages (including error messages) are logged. If this option is not used, error messages are sent to `ice.log`. If you omit this option on Linux computers, error messages will not be logged.
`-o`	Overwrites an existing log file. If this flag is not set, messages are appended to the log file instead.
`-e LDIF_error_log_file`	Specifies a filename where entries that fail are output in LDIF format. This file can be examined, modified to correct the errors, then reapplied to the directory.
`-p URL`	Specifies the location of an XML placement rule to be used by the engine. Placement rules let you change the placement of an entry. See "Conversion Rules" on page 176 for more information.
`-c URL`	Specifies the location of an XML creation rule to be used by the engine. Creation rules let you supply missing information that might be needed to allow an entry to be created successfully on import. For more information, see "Conversion Rules" on page 176.
`-s URL`	Specifies the location of an XML schema mapping rule to be used by the engine. Schema mapping rules let you map a schema element on a source server to a different but equivalent schema element on a destination server. For more information, see "Conversion Rules" on page 176.
`-h` or `-?`	Displays command line help.

Schema Options

The schema options let you use the schema cache to perform schema compare and update operations.

Option	Description
`-C -a`	Updates the destination schema (adds missing schema).
`-C -c filename`	Outputs the destination schema to the specified file.
`-C -n`	Disables schema pre-checking.

Source Handler Options

The source handler option (`-S`) determines the source of the import data. Only one of the following can be specified on the command line.

Option	Description
-SLDIF	Specifies that the source is an LDIF file. For a list of supported LDIF options, see “LDIF Source Handler Options” on page 160.
-SLDAP	Specifies that the source is an LDAP server. For a list of supported LDAP options, see “LDAP Source Handler Options” on page 162
-SDELIM	Specifies that the source is a comma-delimited data file. **NOTE:** For better performance, import data by using OpenText Import Conversion Export utility with LDIF file instead of DELIM. You can use a custom PERL script to generate the output into your desired format. For a list of supported DELIM options, see “DELIM Source Handler Options” on page 165.
-SSCH	Specifies that the source is a schema file. For a list of supported SCH options, see “SCH Source Handler Options” on page 167
-SLOAD	Specifies that the source is a DirLoad template. For a list of supported LOAD options, see “LOAD Source Handler Options” on page 168.

Destination Handler Options

The destination handler option (-D) specifies the destination of the export data. Only one of the following can be specified on the command line.

Option	Description
-DLDIF	Specifies that the destination is an LDIF file. For a list of supported options, see “LDIF Destination Handler Options” on page 161.
-DLDAP	Specifies that the destination is an LDAP server. For a list of supported options, see “LDAP Destination Handler Options” on page 164.
-DDELIM	Specifies that the destination is a comma-delimited file. For a list of supported options, see “DELIM Destination Handler Options” on page 167.

LDIF Source Handler Options

The LDIF source handler reads data from an LDIF file, then sends it to the OpenText Import Conversion Export engine.

Option	Description
`-f LDIF_file`	Specifies a filename containing LDIF records read by the LDIF source handler and sent to the engine. If you omit this option on Linux computers, the input will be taken from stdin.
`-a`	If the records in the LDIF file are content records (that is, they contain no changetypes), they will be treated as records with a changetype of add.
`-c`	Prevents the LDIF source handler from stopping on errors. This includes errors on parsing LDIF and errors sent back from the destination handler. When this option is set and an error occurs, the LDIF source handler reports the error, finds the next record in the LDIF file, then continues.
`-n`	Does not perform update operations, but prints what would be done. When this option is set, the LDIF source handler parses the LDIF file but does not send any records to the OpenText Import Conversion Export engine (or to the destination handler).
`-m`	If the records in the LDIF file are content records (that is, they contain no changetypes), they will be treated as records with a changetype of modify.
`-x`	If the records in the LDIF file are content records (that is, they contain no changetypes), they will be treated as records with a changetype of delete.
`-R value`	Specifies the range of records to be processed.
`-v`	Enables the verbose mode of the handler.
`-e value`	Scheme to be used for decrypting the attribute values present in the LDIF file. [des/3des].
`-E value`	Password for decryption of attributes. You can use the `ADM_E_SRC_PASSWD` variable to encrypt the LDIF source handler.

LDIF Destination Handler Options

The LDIF destination handler receives data from the OpenText Import Conversion Export engine and writes it to an LDIF file.

Option	Description
`-f LDIF_file`	Specifies the filename where LDIF records can be written. If you omit this option on Linux computers, the output will go to stdout.
`-B`	Do not suppress printing of binary values.
`-b`	Do not base64 encode LDIF data.
`-e value`	Scheme to be used for encrypting the attribute values received from the LDAP server.[des/3des].
`-E value`	Password for encryption of attributes. You can use the `ADM_E_DEST_PASSWD` variable to encrypt the LDIF destination handler.

LDAP Source Handler Options

The LDAP source handler reads data from an LDAP server by sending a search request to the server. It then sends the search entries it receives from the search operation to the OpenText Import Conversion Export engine.

Option	Description
`-s server_name`	Specifies the DNS name or IP address of the LDAP server that the handler will send a search request to. The default is the local host. **NOTE:** If you are using the version 9.1 and above, you should specify the FQDN of the LDAP server for this option.
`-p port`	Specifies the integer port number of the LDAP server specified by *server_name*. The default is 389. For secure operations, the default port is 636. When ICE is communicating with an LDAP server on the SSL port (default 636) without a certificate, it chooses to accept any server certificate and assumes it to be a trusted one. This should only be used in controlled environments where encrypted communication between servers and clients is desired but server verification is not necessary.
`-d DN`	Specifies the distinguished name of the entry that should be used when binding to the server-specified bind operation.
`-w password`	Specifies the password attribute of the entry specified by *DN*. You can use the ADM_SRC_PASSWD variable to supply passwords to the LDAP Source handler.
`-W`	Prompts for the password of the entry specified by *DN*. This option is applicable only for Linux.
`-F filter`	Specifies an RFC 1558-compliant search filter. If you omit this option, the search filter defaults to `objectclass=*`.
`-n`	Does not actually perform a search, but shows what search would be performed.
`-a attribute_list`	Specifies a comma-separated list of attributes to retrieve as part of the search. In addition to attribute names, there are three other values: ◆ Get no attributes (`1.1`) ◆ All user attributes (`*`) ◆ An empty list gets all nonoperational attributes If you omit this option, the attribute list defaults to the empty list.
`-o attribute_list`	Specifies a comma-separated list of attributes to be omitted from the search results received from the LDAP server before they are sent to the engine. This option is useful in cases where you want to use a wildcard with the `-a` option to get all attributes of a class and then remove a few of them from the search results before passing the data on to the engine. For example, `-a* -o telephoneNumber` searches for all user-level attributes and filters the telephone number from the results.

Option	Description
`-R`	Specifies to not automatically follow referrals. The default is to follow referrals with the name and password given with the `-d` and `-w` options.
`-e value`	Specifies which debugging flags should be enabled in the LDAP client SDK. For more information, see “Troubleshooting” on page 811.
`-b base_DN`	Specifies the base distinguished name for the search request. If this option is omitted, the base DN defaults to " " (empty string).
`-c search_scope`	Specifies the scope of the search request. Valid values are the following: ◆ `One`: Searches only the immediate children of the base object. ◆ `Base`: Searches only the base object entry itself. ◆ `Sub`: Searches the LDAP subtree rooted at and including the base object. If you omit this option, the search scope defaults to `Sub`.
`-r deref_aliases`	Specifies the way aliases should be dereferenced during the search operation. Values include the following: ◆ `Never`: Prevents the server from dereferencing aliases. ◆ `Always`: Causes aliases to be dereferenced when locating the base object of the search and when evaluating entries that match the search filter. ◆ `Search`: Causes aliases to be dereferenced when applying the filter to entries within the scope of the search after the base object has been located, but not when locating the base object itself. ◆ `Find`: Causes aliases to be dereferenced when locating the base object of the search, but not when actually evaluating entries that match the search filter. If you omit this option, the alias dereferencing behavior defaults to `Never`.
`-l time_limit`	Specifies a time limit (in seconds) for the search.
`-z size _limit`	Specifies the maximum number of entries to be returned by the search.
`-V version`	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
`-v`	Enables verbose mode of the handler.
`-L filename`	Specifies a file in PEM format containing a server key used for SSL authentication with default value `/var/opt/novell/eDirectory/data/SSCert.pem`. **NOTE:** If the LDAP server is using EC Certificates, you should pass `SSECCert.pem` along with this option.
`-A`	Retrieves attribute names only. Attribute values are not returned by the search operation.
`-t`	Prevents the LDAP handler from stopping on errors.

Option	Description
`-m`	LDAP operations will be modifies.
`-x`	LDAP operations will be deletes.
`-k`	This option is no longer supported. To use SSL, specify a valid certificate using the `-L` option.
`-M`	Enables the Manage DSA IT control.
`-MM`	Enables the Manage DSA IT control, and makes it critical.

LDAP Destination Handler Options

The LDAP destination handler receives data from the OpenText Import Conversion Export engine and sends it to an LDAP server in the form of update operations to be performed by the server.

For information about hashed password in an LDIF file, see “Appendix H, “Troubleshooting,” on page 811”.

Option	Description
`-s` *`server_name`*	Specifies the DNS name or IP address of the LDAP server that the handler will send a search request to. The default is the local host.
`-p` *`port`*	Specifies the integer port number of the LDAP server specified by *`server_name`*. The default is 389. For secure operations, the default port is 636.
`-d` *`DN`*	Specifies the distinguished name of the entry that should be used when binding to the server-specified bind operation.
`-w` *`password`*	Specifies the password attribute of the entry specified by *`DN`*. You can use the ADM_DEST_PASSWD variable to supply passwords to the LDAP destination handler.
`-W`	Prompts for the password of the entry specified by *`DN`*. This option is applicable only for Linux.
`-B`	Use this option if you do not want to use asynchronous LDAP Bulk Update/ Replication Protocol (LBURP) requests for transferring update operations to the server. Instead, use standard synchronous LDAP update operation requests. For more information, see “LDAP Bulk Update/Replication Protocol” on page 185.
`-F`	Allows the creation of forward references. When an entry is going to be created before its parent exists, a placeholder called a **forward reference** is created for the entry’s parent to allow the entry to be successfully created. If a later operation creates the parent, the forward reference is changed into a normal entry.

Option	Description
`-l`	Stores password values using the simple password method of the NetIQ Modular Authentication Service (NMAS). Passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers.
`-e value`	Specifies which debugging flags should be enabled in the LDAP client SDK. For more information, see “Appendix H, “Troubleshooting,” on page 811”.
`-V version`	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
`-L filename`	Specifies a file in PEM format containing a server key used for SSL authentication with default value `/var/opt/novell/eDirectory/data/SSCert.pem`. **NOTE:** If the LDAP server is using EC Certificates, you should pass `SSECCert.pem` along with this option.
`-k`	This option is no longer supported. To use SSL, specify a valid certificate using the `-L` option.
`-M`	Enables the Manage DSA IT control.
`-MM`	Enables the Manage DSA IT control, and makes it critical.
`-P`	Enables concurrent LBURP processing. This option is enabled only if all the operations in the LDIF are add. When you use the `-F` option, `-P` is enabled by default.
`-Z`	Specifies the number of asynchronous requests. This indicates the number of entries the ICE client can send to the LDAP server asynchronously before waiting for any result back from the server.

DELIM Source Handler Options

The DELIM source handler reads data from a comma-delimited data file, then sends it to the destination handler.

Option	Description
`-f filename`	Specifies a filename containing comma-delimited records read by the DELIM source handler and sent to the destination handler.
`-F value`	Specifies a file containing the attribute data order for the file specified by `-f`. By default, the number of columns for an attribute in the delimited file equals maximum number of values for the attribute. If an attribute is repeated, the number of columns equals the number of times the attribute repeats in the template. If this option is not specified, enter this information directly using `-t`. See “Performing a Comma-Delimited Import” on page 171 for more information.

Option	Description
`-t value`	The comma-delimited list of attributes specifying the attribute data order for the file specified by `-f`. By default, the number of columns for an attribute in the delimited file equals maximum number of values for the attribute. If an attribute is repeated, the number of columns equals the number of times the attribute repeats in the template. Either this option or `-F` must be specified. See "Performing a Comma-Delimited Import" on page 171 for more information.
`-c`	Prevents the DELIM source handler from stopping on errors. This includes errors on parsing comma-delimited data files and errors sent back from the destination handler. When this option is set and an error occurs, the DELIM source handler reports the error, finds the next record in the comma-delimited data file, then continues.
`-n value`	Specifies the LDAP naming attribute for the new object. This attribute must be contained in the attribute data you specify using `-F` or `-t`.
`-l value`	Specifies the path to append the RDN to (such as `o=myCompany`). If you are passing the DN, this value is not necessary.
`-o value`	Comma-delimited list of object classes (if none is contained in your input file) or additional object classes such as auxiliary classes. The default value is `inetorgperson`.
`-i value`	Comma-delimited list of columns to skip. This value is an integer specifying the number of the column to skip. For example, to skip the third and fifth columns, specify `i3,5`.
`-d value`	Specifies the delimiter. The default delimiter is a comma (,). The following values are special case delimiters: ◆ `[q]` = quote (a single " as the delimiter) ◆ `[t]` = tab For example, to specify a tab as a delimiter, you would pass `-d[t]`.
`-q value`	Specifies the secondary delimiter. The default secondary delimiter is single quotes (' '). The following values are special case delimiters: ◆ `[q]` = quote (a single " as the delimiter) ◆ `[t]` = tab For example, to specify a tab as a delimiter, you would pass `-q[t]`.
`-v`	Runs in verbose mode.
`-k value`	Specifies the first line in the delimited file is the template. If this option is used with `-t` or `-F`, the template specified is checked for consistency with that in the delimited file.

DELIM Destination Handler Options

The DELIM destination handler receives data from the source handler and writes it to a comma-delimited data file.

Option	Description
`-f filename`	Specifies the filename where comma-delimited records can be written.
`-F value`	Specifies a file containing the attribute data order for the file specified by `-f`. By default, the number of columns for an attribute in the delimited file equals maximum number of values for the attribute. If an attribute is repeated, the number of columns equals the number of times the attribute repeats in the template. If this option is not specified, enter this information directly using `-t`.
`-t value`	The comma-delimited list of attributes specifying the attribute data order for the file specified by `-f`. By default, the number of columns for an attribute in the delimited file equals maximum number of values for the attribute. If an attribute is repeated, the number of columns equals the number of times the attribute repeats in the template. Either this option or `-F` must be specified.
`-l value`	Can be either RDN or DN. Specifies whether the driver should place the entire DN or just the RDN in the data. RDN is the default value.
`-d value`	Specifies the delimiter. The default delimiter is a comma (,). The following values are special case delimiters: ◆ `[q]` = quote (a single " as the delimiter) ◆ `[t]` = tab For example, to specify a tab as a delimiter, you would pass `-d[t]`.
`-q value`	Specifies the secondary delimiter. The default secondary delimiter is single quotes (' '). The following values are special case delimiters: ◆ `[q]` = quote (a single " as the delimiter) ◆ `[t]` = tab For example, to specify a tab as a delimiter, you would pass `-q[t]`.
`-n value`	Specifies a naming attribute to be appended during import, for example, `cn`.

SCH Source Handler Options

The SCH handler reads data from a legacy NDS or OpenText eDirectory schema file (files with a `*.sch` extension), then sends it to the OpenText Import Conversion Export engine. You can use this handler to implement schema-related operations on an LDAP Server, such as extensions using a `*.sch` file as input.

The SCH handler is a source handler only. You can use it to import `*.sch` files into an LDAP server, but you cannot export `*.sch` files.

The options supported by the SCH handler are shown in the following table.

Option	Description
`-f filename`	Specifies the full path name of the `*.sch` file.
`-v`	(Optional) Run in verbose mode.

LOAD Source Handler Options

The DirLoad handler generates OpenText eDirectory information from commands in a template. This template file is specified with the `-f` argument and contains the attribute specification information and the program control information.

Option	Description
`-f filename`	Specifies the template file containing all attribute specification and all control information for running the program.
`-c`	Continues to the next record if an error is reported.
`-v`	Runs in verbose mode.
`-r`	Changes the request to a delete request so the data is deleted instead of added. This allows you to remove records that were added using a DirLoad template.
`-m`	Indicates that modify requests will be in the template file.

Attribute Specifications determines the context of new objects.

See the following sample attribute specification file:

```
givenname: $R(first)
initial: $R(initial)
sn: $R(last)
dn:cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn),ou=dev,ou=ds,o=novell
objectclass: inetorgperson
telephonenumber: 1-800-$N(1-999,%03d)-$C(%04d)
title: $R(titles)
locality: Our location
```

The format of the attribute specification file resembles an LDIF file, but allows some powerful constructs to be used to specify additional details and relationships between the attributes.

Unique Numeric Value inserts a numeric value that is unique for a given object into an attribute value.

Syntax: `$C[(<format)]`

The optional *<format* specifies a print format that is to be applied to the value. Note that if no format is specified, the parenthesis cannot be used either:

```
$C
$C(%d)
$C(%04d)
```

The plain $C inserts the current numeric value into an attribute value. This is the same as $C(%d) because "%d" is the default format that the program uses if none was specified. The numeric value is incremented after each object, so if you use $C multiple times in the attribute specification, the value is the same within a single object. The starting value can be specified in the settings file by using the !COUNTER=*value* syntax.

Random Numeric Value inserts a random numeric value into an attribute value using the following syntax:

$N(<*low*-<*high*[,<format])]

<*low* and <*high* specify the lower and upper bounds, respectively, that are used as a random number is generated. The optional <*format* specifies a print format that is to be applied to a value from the list.

```
$N(1-999)
$N(1-999,%d)
$N(1-999,%03d)
```

Random String Value From a List inserts a randomly selected string from a specified list into an attribute value using the following syntax:

$R(<*filename*[,<*format*])]

The <*filename* specifies a file that contains a list of values. This can be an absolute or relative path to a file. Several files containing the lists are included with this package. The values are expected to be separated by a newline character.

The optional <*format* specifies a print format that is to be applied to a value from the list.

```
$A(givenname)
$A(givenname,%s)
$A(givenname,%.1s)
```

It is important to note that no forward references are allowed. Any attribute whose value you are going to use must precede the current attribute in the attribute specification file. In the example below, the cn as part of the DN is constructed from givenname, initial, and sn. Therefore, these attributes must precede the DN in the settings file.

```
givenname: $R(first)
initial: $R(initial)
sn: $R(last)
dn:o=novell,ou=dev,ou=ds,cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn)
```

The DN receives special handling in the LDIF file: no matter what the location of DN is in the settings, it will be written first (as per LDIF syntax) to the LDIF file. All other attributes are written in the order they appear.

Control Settings provide some additional controls for the object creation. All controls have an exclamation point (!) as the first character on the line to separate them from attribute settings. The controls can be placed anywhere in the file.

```
!COUNTER=300
!OBJECTCOUNT=2
!CYCLE=title
!UNICYCLE=first,last
!CYCLE=ou,BLOCK=10
```

- Counter

  Provides the starting value for the unique counter value. The counter value is inserted to any attribute with the $C syntax.

- Object Count

  OBJECTCOUNT determines how many objects are created from the template.

- Cycle

  CYCLE can be used to modify the behavior of pulling random values from the files ($R-syntax). This setting has three different values.

  ```
  !CYCLE=title
  ```

  Anytime the list named “title” is used, the next value from the list is pulled rather than randomly selecting a value. After all values have been consumed in order, the list starts from the beginning again.

  ```
  !CYCLE=ou,BLOCK=10
  ```

  Each value from list “ou” is to be used 10 times before moving to the next value.

  The most interesting variant of the CYCLE control setting is UNICYCLE. It specifies a list of sources that are cycled through in left-to-right order, allowing you to create guaranteed unique values if desired. If this control is used, the OBJECTCOUNT control is used only to limit the number of objects to the maximum number of unique objects that can be created from the lists. In other words, if the lists that are part of UNICYCLE can produce 15000 objects, then OBJECTCOUNT can be used to reduce that number, but not to increase it.

  For example, assume that the givenname file contains two values (Doug and Karl) and the sn file contains three values (Hoffman, Schultz, and Grieger).With the control setting !UNICYCLE=givenname,sn and attribute definition cn: $R(givenname) $R(sn), the following cns are created:

  ```
  cn: Doug Hoffmancn cn: Karl Hoffmancn cn: Doug Schultzcn cn: Karl
  Schultzcn cn: Doug Griegercn cn: Karl Grieger
  ```

Examples

Listed below are sample commands that can be used with the OpenText Import Conversion Export command line utility for the following functions:

- “Performing an LDIF Import” on page 171
- “Performing an LDIF Export” on page 171
- “Performing a Comma-Delimited Import” on page 171
- “Performing a Comma-Delimited Export” on page 172
- “Performing a Data Migration between LDAP Servers” on page 172
- “Performing a Schema Import” on page 173

- "Performing a LOAD File Import" on page 173
- "Performing an LDIF Export from LDAP server having encrypted attributes" on page 176
- "Performing an LDIF Import having encrypted attributes" on page 176

Performing an LDIF Import

To perform an LDIF import, combine the LDIF source and LDAP destination handlers, for example:

```
ice -S LDIF -f entries.ldif -D LDAP -s server1.acme.com -p 389 -d
cn=admin,c=us -w secret
```

This command reads LDIF data from `entries.ldif` and sends it to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us, and the password "secret."

Performing an LDIF Export

To perform an LDIF export, combine the LDAP source and LDIF destination handlers. For example:

```
ice -S LDAP -s server1.acme.com -p 389 -d cn=admin,c=us -w password -F
objectClass=* -c sub -D LDIF -f server1.ldif
```

This command performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us and the password "password" and outputs the data in LDIF format to `server1.ldif`.

Performing a Comma-Delimited Import

To perform a comma-delimited import, use a command similar to the following:

```
ice -S DELIM -f/tmp/in.csv -F /tmp/order.csv -ncn -lo=acme -D LDAP -s
server1.acme.com -p389 -d cn=admin,c=us -w secret
```

This command reads comma-delimited values from the `/tmp/in.csv` file and reads the attribute order from the `/tmp/order.csv` file. For each attribute entry in `in.csv`, the attribute type is specified in `order.csv`. For example, if `in.csv` contains

```
pat,pat,engineer,john
```

then `order.csv` would contain

```
dn,cn,title,sn
```

The information in `order.csv` could be input directly using the `-t` option.

The data is then sent to the LDAP server `server1.acme.com` at port 389 using the identity `cn=admin,c=us`, and password "`secret`".

This example specifies that cn should become the new DN for this object using the `-n` option, and this object was added to the organization container acme using the `-l` option.

Comma-delimited files generated using OpenText Import Conversion Export utility have the template used for generating them in the first line. To specify that first line in the delimited file is the template, use the `-k` option. If `-F` or `-t` is used with `-k`, the template specified should be consistent with that in the delimited file, where both have exactly the same attributes. However, the number of occurrences and the order of appearance of each attribute can differ. In the above example, `in.csv` contains

dn,cn,title,title,title,sn in the first line. The following templates are consistent and can be used with -t or -F when -k is used:

dn,cn,title,sn (number of repetitions of attribute title differs)

dn,sn,title,cn (order of attributes differ)

However, the following are not consistent with the template in in.csv and hence cannot be specified with -t or -F when -k is used:

dn,cn,title,sn,objectclass (new attribute objectclass)

dn,cn,title (missing attribute sn)

Performing a Comma-Delimited Export

To perform a comma-delimited export, use a command similar to the following:

```
ice -S LDAP -s server1.acme.com -p 389 -d cn=admin,c=us -w password -F
objectClass=* -c sub -D DELIM -f /tmp/server1.csv -F order.csv
```

This command performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us and the password “password” and outputs the data in comma-delimited format to the /tmp/server1.csv file.

If any attribute in the order.csv has multiple values, /tmp/server1.csv, the number of columns for this attribute equals maximum number of values for the attribute. If an attribute repeats in order.csv, the number of columns for this attribute equals the number of times the attribute repeats.

For example, if order.csv contains dn,sn,objectclass, and objectclass has 4 values, whereas dn and sn have only 1 value for all the entries exported, dn and sn would have 1 column each, whereas objectclass would have 4 columns. If you want only 2 values for objectclass to be output to the delimited file, order.csv should contain dn,sn,objectclass,objectclass.

In both cases the attributes are written to the /tmp/server1.csv in the first line. In the first case, /tmp/server1.csv would have dn,sn,objectclass,objectclass,objectclass,objectclasss in the first line of /tmp/server1.csv, and in the second case, it would have dn,sn,objectclass,objectclass.

To prevent the first line to be treated as a sequence of attributes during a subsequent import, use the -k option. See “Performing a Comma-Delimited Import” on page 171 for more information.

Performing a Data Migration between LDAP Servers

To perform a data migration between LDAP servers, combine the LDAP source and LDAP destination handlers. For example:

```
ice -S LDAP -s server1.acme.com -p 389 -d cn=admin,c=us -w password -F
objectClass=* -c sub -D LDAP -s server2.acme.com -p 389 -d cn=admin,c=us -
w secret
```

This command performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us and the password “password” and sends it to the LDAP server server2.acme.com at port 389 using the identity cn=admin,c=us and the password “secret.”

Performing a Schema Import

To perform a schema file import, use a command similar to the following:

```
ice -S SCH -f $HOME/myfile.sch -D LDAP -s myserver -d cn=admin,o=novell -w
passwd
```

This command reads schema data from `myfile.sch` and sends it to the LDAP server myserver using the identity `cn=admin,o=novell` and the password "`passwd`."

Performing a LOAD File Import

To perform a LOAD file import, use a command similar to the following:

```
ice -S LOAD -f attrs -D LDIF -f new.ldf
```

In this example, the contents of the attribute file `attrs` is as follows:

```
#=====================================================================

#  DirLoad 1.00
#=====================================================================

!COUNTER=300

!OBJECTCOUNT=2
#---------------------------------------------------------------------

#  ATTRIBUTE TEMPLATE
# ---------------------------------------------------------------------

objectclass: inetorgperson

givenname: $R(first)

initials: $R(initial)

sn: $R(last)

dn: cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn),ou=$R(ou),ou=dev,o=novell,

telephonenumber: 1-800-$N(1-999,%03d)-$C(%04d)

title: $R(titles)
```

Running the previous command from a command prompt produces the following LDIF file:

```
version: 1

dn: cn=JohnBBill,ou=ds,ou=dev,o=novell

changetype: add

objectclass: inetorgperson

givenname: John

initials: B
```

```
sn: Bill

telephonenumber: 1-800-290-0300

title: Amigo


dn: cn=BobJAmy,ou=ds,ou=dev,o=novell

changetype: add

objectclass: inetorgperson

givenname: Bob

initials: J

sn: Amy

telephonenumber: 1-800-486-0301

title: Pomo
```

Running the following command from a command prompt sends the data to an LDAP server via the LDAP Handler:

```
ice -S LOAD -f attrs -D LDAP -s www.novell.com -d cn=admin,o=novell -w
admin
```

If the previous template file is used, but the following command is used, all of the records that were added with the above command will be deleted.

```
ice -S LOAD -f attrs -r -D LDAP -s www.novell.com -d cn=admin,o=novell -w
admin
```

If you want to use -m to modify, the following is an example of how to modify records:

```
# ====================================================================

#  DirLoad 1.00

# ====================================================================
!COUNTER=300

!OBJECTCOUNT=2

#--------------------------------------------------------------------

#  ATTRIBUTE TEMPLATE

# --------------------------------------------------------------------

dn: cn=$R(first),%.1s)($R(initial),%.1s)$R(last),ou=$R(ou),ou=dev,o=novell

delete: givenname

add: givenname

givenname: test1
```

```
replace: givenname
givenname: test2
givenname: test3
```

If the following command is used where the `attrs` file contains the data above:

```
ice -S LOAD -f attrs -m -D LDIF -f new.ldf
```

then the results would be the following LDIF data:

```
version: 1
dn: cn=BillTSmith,ou=ds,ou=dev,o=novell
changetype: modify
delete: givenname
-
add: givenname
givenname: test1
-
replace: givenname
givenname: test2
givenname: test3
-
dn: cn=JohnAWilliams,ou=ldap,ou=dev,o=novell
changetype: modify
delete: givenname
-
add: givenname
givenname: test1
-
replace: givenname
givenname: test2
givenname: test3
-
```

Performing an LDIF Export from LDAP server having encrypted attributes

To perform an LDIF export from LDAP server having encrypted attributes, combine the LDAP source and LDIF destination handlers along with the scheme and password for encryption, for example:

```
ice -S LDAP -s server1.acme.com -p 636 -L cert-server1.pem -d cn=admin,c=us
-w password -F objectClass=* -c sub -D LDIF -f server1.ldif -e des -E
secret
```

Performing an LDIF Import having encrypted attributes

To perform an LDIF import of a file having attributes encrypted by ICE previously, combine the LDIF source with the scheme and password used previously for exporting the file and LDAP destination handlers, for example:

```
ice -S LDIF -f server1.ldif -e des -E secret -D LDAP -s server2.acme.com -
p 636 -L cert-server2.pem -d cn=admin,c=us -w password
```

Conversion Rules

The OpenText Import Conversion Export engine lets you specify a set of rules that describe processing actions to be taken on each record received from the source handler and before the record is sent on to the destination handler. These rules are specified in XML (either in the form of an XML file or XML data stored in the directory) and solve the following problems when importing entries from one LDAP directory to another:

- Missing information
- Hierarchical differences
- Schema differences

There are three types of conversion rules:

Rule	Description
Placement	Changes the placement of an entry. For example, if you are importing a group of users in the l=San Francisco, c=US container but you want them to be in the l=Los Angeles, c=US container when the import is complete, you could use a placement rule to do this. For information on the format of these rules, see “Placement Rules” on page 181.
Creation	Supplies missing information that might be needed to allow an entry to be created successfully on import. For example, assume that you have exported LDIF data from a server whose schema requires only the cn (commonName) attribute for user entries, but the server that you are importing the LDIF data to requires both the cn and sn (surname) attributes. You could use the creation rule to supply a default sn value, (such as " ") for each entry, as it is processed by the engine. When the entry is sent to the destination server, it will have the required sn attribute and can be added successfully. For information on the format of these rules, see “Create Rules” on page 179.

Rule	Description
Schema Mapping	If, when you are transferring data between servers (either directly or using LDIF), there are schema differences in the servers, you can use Schema Mapping to ◆ Extend the schema on the destination server to accommodate the object classes and attribute types in entries coming from the source server. ◆ Map a schema element on the source server to a different but equivalent schema element on the destination server. For information on the format of these rules, see "Schema Mapping Rules" on page 178.

You can enable conversion rules in both the OpenText eDirectory Import/Export Wizard and the command line interface. For more information on XML rules, see "Using XML Rules" on page 177.

Using the Command Line Interface

You can enable conversion rules with the `-p`, `-c`, and `-s` general options on the OpenText Import Conversion Export executable. For more information, see "General Options" on page 159.

Option	Description
`-p` *URL*	Location of an XML placement rule to be used by the engine.
`-c` *URL*	Location of an XML creation rule to be used by the engine.
`-s` *URL*	Location of an XML schema mapping rule to be used by the engine.

For all three options, *URL* must be one of the following:

- A URL of the following format:

  `file://[`*`path/`*`]`*`filename`*

  The file must be on the local file system.
- An RFC 2255-compliant LDAP URL that specifies a base-level search and an attribute list consisting of a single attribute description for a singled-valued attribute type.

Using XML Rules

The OpenText Import Conversion Export conversion rules use the same XML format as OpenText Identity Manager. For more information on OpenText Identity Manager, see *OpenText Identity Manager documentation site* (https://www.netiq.com/documentation/identity-manager-4.10/).

Schema Mapping Rules

The <attr-name-map> element is the top-level element for the schema mapping rules. Mapping rules determine how the import schema interacts with the export schema. They associate specified import class definitions and attributes with corresponding definitions in the export schema.

Mapping rules can be set up for attribute names or class names.

- For an attribute mapping, the rule must specify that it is an attribute mapping, a name space (nds-name is the tag for the source name), the name in the OpenText eDirectory name space, then the other name space (app-name is the tag for the destination name) and the name in that name space. It can specify that the mapping applies to a specific class or it can be applied to all classes with the attribute.
- For a class mapping, the rule must specify that it is a class mapping rule, a name space (OpenText eDirectory or the application), the name in that name space, then the other name space and the name in that name space.

The following is the formal DTD definition of schema mapping rules:

```
<!ELEMENT attr-name-map (attr-name | class-name)*>

<!ELEMENT attr-name (nds-name, app-name)>
<!ATTLIST attr-name
          class-name    CDATA    #IMPLIED>

<!ELEMENT class-name (nds-name, app-name)>

<!ELEMENT nds-name (#PCDATA)>

<!ELEMENT app-name (#PCDATA)>
```

You can have multiple mapping elements in the file. Each element is processed in the order that it appears in the file. If you map the same class or attribute more than once, the first mapping takes precedence.

The following examples illustrate how to create a schema mapping rule.

Schema Rule 1: The following rule maps the source's surname attribute to the destination's sn attribute for the inetOrgPerson class.

```
<attr-name-map>
   <attr-name class-name="inetOrgPperson">
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
</attr-name-map>
```

Schema Rule 2: The following rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
   </class-name>
</attr-name-map>
```

Schema Rule 3: The following example contains two rules. The first rule maps the source's Surname attribute to the destination's sn attribute for all classes that use these attributes. The second rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <attr-name>
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
   </class-name>
</attr-name-map>
```

Example Command: If the schema rules are saved to an `sr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `outt1.ldf`.

```
ice -o -sfile://sr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

Create Rules

Create rules specify the conditions for creating a new entry in the destination directory. They support the following elements:

- **Required Attributes** specifies that an add record must have values for all of the required attributes, or else the add fails. The rule can supply a default value for a required attribute. If a record does not have a value for the attribute, the entry is given the default value. If the record has a value, the record value is used.
- **Matching Attributes** specifies that an add record must have the specific attributes and match the specified values, or else the add fails.
- **Templates** specifies the distinguished name of a Template object in OpenText eDirectory. The OpenText Import Conversion Export utility does not currently support specifying templates in create rules.

The following is the formal DTD definition for create rules:

```
<!ELEMENT create-rules (create-rule)*>

<!ELEMENT create-rule (match-attr*,
                       required-attr*,
                       template?) >
<!ATTLIST create-rule
          class-name    CDATA   #IMPLIED
          description   CDATA   #IMPLIED>

<!ELEMENT match-attr    (value)+ >
<!ATTLIST match-attr
          attr-name      CDATA    #REQUIRED>

<!ELEMENT required-attr (value)*>
<!ATTLIST required-attr
          attr-name     CDATA   #REQUIRED>

<!ELEMENT template EMPTY>
<!ATTLIST template
          template-dn   CDATA   #REQUIRED>
```

You can have multiple create rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped and the skipping does not generate an error.

The following examples illustrate how to format create rules.

Create Rule 1: The following rule places three conditions on add records that belong to the inetOrgPerson class. These records must have givenName and Surname attributes. They should have an L attribute, but if they don't, the create rule supplies a default value of Provo for them.

```
<create-rules>
   <create-rule class-name="inetOrgPerson">
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 2: The following create rule places three conditions on all add records, regardless of their base class:

- The record must contain a givenName attribute. If it doesn’t, the add fails.
- The record must contain a Surname attribute. If it doesn’t, the add fails.
- The record must contain an L attribute. If it doesn't, the attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="Surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 3: The following create rule places two conditions on all records, regardless of base class:

- The rule checks to see if the record has a uid attribute with a value of ratuid. If it doesn't, the add fails.
- The rule checks to see if the record has an L attribute. If it does not have this attribute, the L attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
      <match-attr attr-name="uid">
         <value>cn=ratuid</value>
      </match-attr>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Example Command: If the create rules are saved to an `crl.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `outt1.ldf`.

```
ice -o -cfile://cr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

Placement Rules

Placement rules determine where an entry is created in the destination directory. They support the following conditions for determining whether the rule should be used to place an entry:

- **Match Class:** If the rule contains any match class elements, an objectClass specified in the record must match the class-name attribute in the rule. If the match fails, the placement rule is not used for that record.
- **Match Attribute:** If the rule contains any match attribute elements, the record must contain an attribute value for each of the attributes specified in the match attribute element. If the match fails, the placement rule is not used for that record.
- **Match Path:** If the rule contains any match path elements, a portion of the record's DN must match the prefix specified in the match path element. If the match fails, the placement rule is not used for that record.

The last element in the rule specifies where to place the entry. The placement rule can use zero or more of the following:

- **PCDATA** uses parsed character data to specify the DN of a container for the entries.
- **Copy the Name** specifies that the naming attribute of the old DN is used in the entry's new DN.
- **Copy the Attribute** specifies the naming attribute to use in the entry's new DN. The specified naming attribute must be a valid naming attribute for the entry's base class.
- **Copy the Path** specifies that the source DN should be used as the destination DN.
- **Copy the Path Suffix** specifies that the source DN, or a portion of its path, should be used as the destination DN. If a match-path element is specified, only the part of the old DN that does not match the prefix attribute of the match-path element is used as part of the entry's DN.

The following is the formal DTD definition for the placement rule:

```
<!ELEMENT placement-rules (placement-rule*)>
<!ATTLIST placement-rules
          src-dn-format    (%dn-format;)   "slash"
          dest-dn-format   (%dn-format;)   "slash"
          src-dn-delims    CDATA           #IMPLIED
          dest-dn-delims   CDATA           #IMPLIED>

<!ELEMENT placement-rule (match-class*,
                          match-path*,
                          match-attr*,
                          placement)>
<!ATTLIST placement-rule
          description      CDATA            #IMPLIED>

<!ELEMENT match-class      EMPTY>
<!ATTLIST match-class
          class-name       CDATA            #REQUIRED>

<!ELEMENT match-path       EMPTY>
<!ATTLIST match-path
          prefix           CDATA            #REQUIRED>

<!ELEMENT match-attr       (value)+ >
<!ATTLIST match-attr
          attr-name        CDATA            #REQUIRED>

<!ELEMENT placement        (#PCDATA |
                          copy-name |
                          copy-attr |
                          copy-path |
                          copy-path-suffix)* >
```

You can have multiple placement-rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped and the skipping does not generate an error.

The following examples illustrate how to format placement rules. The `scr-dn-format="ldap"` and `dest-dn-format="ldap"` attributes set the rule so that the name space for the DN in the source and destination is LDAP format.

The OpenText Import Conversion Export utility supports source and destination names only in LDAP format.

Placement Example 1: The following placement rule requires that the record have a base class of inetOrgPerson. If the record matches this condition, the entry is placed immediately subordinate to the test container and the left-most component of its source DN is used as part of its DN.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-class class-name="inetOrgPerson"></match-class>
       <placement>cn=<copy-name/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with a base class of inetOrgPerson and with the following DN:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
```

would have the following DN in the destination directory:

```
dn: cn=Kim Jones, o=test
```

Placement Example 2: The following placement rule requires that the record have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and the left-most component of its source DN is used as part of its DN.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement>cn=<copy-name/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following DN in the destination directory:

```
dn: cn=Kim Jones, o=test
```

Placement Example 3: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and its sn attribute is used as part of its DN. The specified attribute in the copy-attr element must be a naming attribute of the entry's base class.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
  <placement-rule>
    <match-attr attr-name="sn"></match-attr>
    <placement>cn=<copy-attr attr-name="sn"/>,o=test</placement>
  </placement-rule>
</placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following DN in the destination directory:

```
dn: cn=Jones, o=test
```

Placement Example 4: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the source DN is used as the destination DN.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement><copy-path/></placement>
    </placement-rule>
 </placement-rules>
```

Placement Example 5: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry's entire DN is copied to the test container.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement><copy-path-suffix/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
```

would have the following DN in the destination directory:

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ, o=test
```

Placement Example 6: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry's entire DN is copied to the neworg container.

```
<placement-rules>
 <placement-rule>
  <match-path prefix="o=engineering"/>
  <placement><copy-path-suffix/>o=neworg</placement>
 </placement-rule>
</placement-rules>
```

For example:

```
dn: cn=bob,o=engineering
```

becomes

```
dn: cn=bob,o=neworg
```

Example Command: If the placement rules are saved to a `pr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `foutt1.ldf`.

```
ice -o -pfile://pr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

LDAP Bulk Update/Replication Protocol

The OpenText Import Conversion Export utility uses the LDAP Bulk Update/Replication Protocol (LBURP) to send asynchronous requests to an LDAP server. This guarantees that the requests are processed in the order specified by the protocol and not in an arbitrary order influenced by multiprocessor interactions or the operating system's scheduler.

LBURP also lets the OpenText Import Conversion Export utility send several update operations in a single request and receive the response for all of those update operations in a single response. This adds to the network efficiency of the protocol.

LBURP works as follows:

1. The OpenText Import Conversion Export utility binds to an LDAP server.
2. The server sends a bind response to the client.
3. The client sends a start LBURP extended request to the server.
4. The server sends a start LBURP extended response to the client.
5. The client sends zero or more LBURP operation extended requests to the server.

   These requests can be sent asynchronously. Each request contains a sequence number identifying the order of this request relative to other requests sent by the client over the same connection. Each request also contains at least one LDAP update operation.
6. The server processes each of the LBURP operation extended requests in the order specified by the sequence number and sends an LBURP operation extended response for each request.
7. After all of the updates have been sent to the server, the client sends an end LBURP extended request to the server.
8. The server sends an end LBURP extended response to the client.

The LBURP protocol lets OpenText Import Conversion Export present data to the server as fast as the network connection between the two will allow. If the network connection is fast enough, this lets the server stay busy processing update operations 100% of the time because it never has to wait for OpenText Import Conversion Export to give it more work to do.

The LBURP processor in OpenText eDirectory also commits update operations to the database in groups to gain further efficiency in processing the update operations. LBURP can greatly improve the efficiency of your LDIF imports over a traditional synchronous approach.

LBURP is enabled by default, but you can choose to disable it during an LDIF import.

You can use the command line option to enable or disable LBURP during an LDIF import. For more information, see "-B" on page 164.

Improving the Speed of LDIF Imports

In cases where you have thousands or even millions of records in a single LDIF file you are importing, consider the following:

- "Importing Directly to a Server with a Read/Write Replica" on page 186
- "Using LBURP" on page 186
- "Configuring the Database Cache" on page 186

- "Using Simple Passwords" on page 186
- "Using Indexes Appropriately" on page 187

Importing Directly to a Server with a Read/Write Replica

If it's possible to do so, select a destination server for your LDIF import that has read/write replicas containing all the entries represented in the LDIF file. This will maximize network efficiency.

Avoid having the destination server chain to other OpenText eDirectory servers for updates. This can severely reduce performance. However, if some of the entries to be updated are only on OpenText eDirectory servers that are not running LDAP, you might need to allow chaining to import the LDIF file.

For more information on replicas and partition management, see Chapter 6, "Managing Partitions and Replicas," on page 143.

Using LBURP

Import Conversion Export maximizes network and OpenText eDirectory server processing efficiency by using LBURP to transfer data between the wizard and the server. Using LBURP during an LDIF import greatly improves the speed of your LDIF import.

For more information on LBURP, see "LDAP Bulk Update/Replication Protocol" on page 185.

Configuring the Database Cache

The amount of database cache available for use by OpenText eDirectory has a direct bearing on the speed of LDIF imports, especially as the total number of entries on the server increases. When doing an LDIF import, you might want to allocate the maximum memory possible to OpenText eDirectory during the import. After the import is complete and the server is handling an average load, you can restore your previous memory settings. This is particularly important if the import is the only activity taking place on the OpenText eDirectory server.

For more information on configuring the OpenText eDirectory database cache, see Chapter 19, "Maintaining OpenText eDirectory," on page 529.

Using Simple Passwords

OpenText eDirectory uses public and private key pairs for authentication. Generating these keys is a very CPU-intensive process. With the version 8.7.3 onwards, you can choose to store passwords using the simple password feature of OpenText Modular Authentication Service (NMAS). When you do this, passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers. This greatly improves the speed for loading an object that has password information.

You can use the command line version of the OpenText Import Conversion Export utility to perform LDIF import as explained at: "Using the Command Line Interface" on page 158.

If you choose to store passwords using simple passwords, you must use an NMAS-aware Novell Client to log in to the OpenText eDirectory tree and access traditional file and print services. NMAS must also be installed on the server. LDAP applications binding with name and password will work seamlessly with the simple password feature.

For more information on NMAS, see the Chapter 24, “Understanding OpenText eDirectory’s Authentication Framework,” on page 603.

Using Indexes Appropriately

Having unnecessary indexes can slow down your LDIF import because each defined index requires additional processing for each entry having attribute values stored in that index. You should make sure that you don’t have unnecessary indexes before you do an LDIF import, and you might want to consider creating some of your indexes after you have finished loading the data reviewed predicate statistics to see where they are really needed.

For more information on tuning indexes, see “Index Manager” on page 187.

Index Manager

Index Manager is an attribute of the Server object that lets you manage database indexes. These indexes are used by OpenText eDirectory to significantly improve query performance.

OpenText eDirectory ships with a set of indexes that provide basic query functionality. These default indexes are for the following attributes:

CN	Aliased Object Name
dc	Obituary
Given Name	Member
Surname	Reference
uniqueID	Equivalent to Me
GUID	NLS: Common Certificate
cn_SS	Revision
uniqueID_SS	extensionInfo
ldapAttributeList	ldapClassList

You can also create customized indexes to further improve OpenText eDirectory performance in your environment. For example, if your organization has implemented a new LDAP application that looks up an attribute not indexed by default, it might be useful to create an index for that attribute.

NOTE: Although indexes improve search performance, additional indexes also add to directory update time. As a general rule, create new indexes only if you suspect performance issues are related to a particular directory lookup.

Using OpenText Identity Console, you can create or delete indexes. You can also view and manage the properties of an index, including the index name, state, type, rule, and attribute indexed.

Creating an Index

1. On the OpenText Identity Console home page, click the **Index Management** tile.
2. Click **Create Index** +.
3. On the **Create Index** page enter the name at the **Name** field.
4. From **Select server** list > select server.
5. From **Select Attribute** list > select attribute.
6. Select the index rule.
   - **Value** matches the entire value or the first part of the value of an attribute. For example, value matching could be used to find entries with a LastName that is equal to “Jensen” and entries with a LastName that begins with “Jen.”
   - **Presence** requires only the presence of an attribute rather than specific attribute values. A query to find all entries with a Login Script attribute would use a presence index.
   - **Substring** matches a subset of the attribute value string. For example, a query to find a LastName with “der” would return matches for Derington, Anderson, and Lauder.

     A substring index is the most resource-intensive index to create and maintain.
7. Click **Create**.

   The **Index Created successfully on 1 server** massage appears.

   Restart Limber as a background process and initiate the change.

IMPORTANT: The $ character is used as a delimiter for attribute values. If you use the $ character in your index name, you must use a preceding backslash (\) character to escape the $ character when working with indexes via LDAP.

Deleting an Index

Indexes might outlive their usefulness. You can delete user-defined and auto-created indexes that are no longer a benefit.

1. On the OpenText Identity Console home page, click the **Index Management** tile.
2. On the Index page, select the user- or auto-added index you want to delete.
3. Click **Delete**  to update the index table.
4. On the **Delete Confirm** popup > click **OK** to confirm delete.

   Restart Limber as a background process and initiate the change.

Taking an Index Offline

During peak times you might want to tune performance by temporarily taking indexes offline. For example, to achieve additional bulk-load speed, you might want to suspend all of the user-defined indexes. Because each object addition or modification requires updating defined indexes, having all indexes active might slow down bulk-loading of data. After the bulk-load is completed, the indexes can be brought online again.

1. On the OpenText Identity Console home page, click the **Index Management** tile.
2. On the **Index** page, select the check box against the index that you want to take off-line, click **State** drop-down menu > select **Offline**.

   The index state changes from Online to Offline in the display table. An index can be in any of the following states:

   - **Online** : Currently running.
   - **Offline** : Suspended. The index can be started again by clicking **Bring Online**.
   - **New** : Waiting to move to Online.
   - **Deleted** : Waiting to be removed from the index table.

Managing Indexes on Other Servers

If you've found a particular index to be useful on one server and you see the need for this index on another server, you can copy the index definition from one server to another. In reviewing predicate data, you might also find just the opposite case: an index that was meeting a need for several servers is no longer useful on one of these servers. In that case, you could delete the index from the single server that isn't benefitting from the index.

Index Manager allows you to target a single instance of an index without impacting all instances.

1. On the OpenText Identity Console home page, click the **Index Management** tile.
2. On the **Index** page, select the index that you want to modify.

   When you select an index, servers in the tree providing that index are listed.
3. On the **Modify Index** page select additional servers.
4. Click **Save**.

   The **Index modified successfully!** message appears.

OpenText eDirectory Service Manager

The OpenText eDirectory Service Manager provides information about available OpenText eDirectory services and their states. You can also use the Service Manager to start and stop these services.

Service Manager manages only OpenText eDirectory services. This is done with the help of the `dsservcfg.xml` configuration file, which lists the services to be managed on various platform. It also lets you add or remove services from the list.

You can access the OpenText eDirectory Service Manager through the following methods:

- "Using the Client Service Manager eMTool" on page 190

Using the Client Service Manager eMTool

The OpenText eDirectory Management Toolbox (eMBox) Client is a command line Java client that gives you remote access to the OpenText eDirectory Service Manager eMTool. The `emboxclient.jar` file is installed on your server as part of OpenText eDirectory. You can run it on any machine with a JVM. For more information on the Client, see "Using the Command Line Client" on page 560.

To use the Client Service Manager eMTool:

1. Run the Client in interactive mode by entering the following at the command line:

   ```
   java -cp path_to_the_file/emboxclient.jar  -i
   ```

   (If you have already put the `emboxclient.jar` file in your class path, you only need to enter `java -i`.)

   The Client prompt appears:

   ```
   Client>
   ```

2. Log in to the server that will run Service Manager by entering the following:

   ```
   login -s server_name_or_IP_address -p port_number
   -u username.context -w password -n
   ```

   The port number is usually 80 or 8028, unless you have a Web server that is already using the port. The `-n` option opens a nonsecure connection.

   The Client indicates whether the login is successful.

3. Enter one of the following Service Manager commands:

Command	Description
`service.serviceList`	Lists the available OpenText eDirectory services.
`service.serviceStart -n Module_name`	Starts the specified OpenText eDirectory service.
`service.serviceStop -n Module_name`	Stops the specified OpenText eDirectory service.
`service.serviceInfo -n Module_name`	Displays information for the specified service.

   You can also use the `list -t service` command in the Client to list the Service Manager options with details. See "Listing eMTools and Their Services" on page 563 for more information.

4. Log out from the Client by entering the following command:

   ```
   logout
   ```

5. Exit the Client by entering the following command:

   ```
   exit
   ```

Offline Bulkload Utility

ldif2dib utility lets you bulkload data from LDIF files to the OpenText eDirectory database (DIB), when the OpenText eDirectory server is offline. OpenText eDirectory supports this utility on both Linux and Windows platforms. This is an offline utility and achieves faster bulkloads compared to the other online tools. The utility uses the existing directory and does not create a new database while importing entries from an LDIF file to the DIB.

ldif2dib utility is needed when you need to populate a large user database with entries from an LDIF file. Online tools such as ICE or ldapmodify are slower compared to ldif2dib due to overheads associated with online bulk load such as schema checking, protocol translation, and access control checks. ldif2dib allows for fast up time when a large user database needs to be populated and when initial down time is not an issue.

Improving Bulkload Performance

OpenText eDirectory provides you with new options to increase the bulkload performance.

The following are the tunable parameters for bulkload performance using the OpenText Import Convert Export (ICE) utility.

- "OpenText eDirectory Cache Settings" on page 191
- "LBURP Transaction Size Setting" on page 191
- "Increasing the Number of Asynchronous Requests in ICE" on page 192
- "Increased Number of LDAP Writer Threads" on page 192
- "Disabling Schema Validation in ICE" on page 193
- "Backlinker" on page 193
- "Disabling ACL Templates" on page 193
- "Enabling/Disabling Inline Cache" on page 195
- "Increasing the LBURP Time Out Period" on page 195

Also refer to the various operating system tunable parameters.

OpenText eDirectory Cache Settings

To optimize the bulkload performance, allocate a higher percentage of the OpenText eDirectory cache for block cache.

For more details refer to "Tuning OpenText eDirectory Subsystems".

LBURP Transaction Size Setting

The LBURP transaction size sets the number of records that are sent from ICE to the LDAP server during a single transaction. Increasing this value can improve bulkload performance, assuming that you have adequate memory and that the increase does not cause I/O contention.

The default transaction size is 25, which is appropriate for small LDIF files (fewer than 100,000 operations) but not for a large number of records. The LBURP transaction size can be set between 1 and 350.

Modifying the Transaction Size

To modify the transaction size, modify the required value for the n4u.ldap.lburp.transize parameter in `/etc/opt/novell/eDirectory/conf/nds.conf`. In ideal scenarios, a higher transaction size ensures faster performance. However, the transaction size must not be set to arbitrarily high values for the following reasons:

- A larger transaction size requires the server to allocate more memory to process the transaction. If the system is running low on memory, this can cause a slowdown due to swapping.
- The LDIF file should be free of errors and any entries already existing in OpenText eDirectory should be commented out. Even if a single error exists in the transaction (including cases where the object to be added already exists in the directory), OpenText eDirectory ignores the LBURP transaction setting and performs a commit after each operation to ensure data integrity.

  For more information, see Appendix H, “Troubleshooting,” on page 811.
- LBURP optimization works only for leaf objects. If the transaction contains both a container and its subordinate objects, OpenText eDirectory treats this as an error. To avoid this, we recommend loading the container objects first using a separate LDIF file or enables the use of forward references.

  For more information, see “Enabling Forward References” in the Appendix H, “Troubleshooting,” on page 811.

Increasing the Number of Asynchronous Requests in ICE

This refers to the number of entries the ICE client can send to the LDAP server asynchronously before waiting for any result back from the server.

The number of asynchronous requests can be set between 10 and 200. The default value is 100. Any value less than the minimum value (10) would fallback to the default. The minimum value is appropriate for small LDIF files.

In ideal scenarios, a higher window size ensures faster performance. However, the window size must not be set to arbitrarily high values because a larger window size requires the client to allocate more memory to process the entries in the LDIF file. If the system is running low on memory, this can cause a slowdown due to swapping.

You can modify the number of asynchronous requests in ICE using the ICE command line option.

Using ICE Command Line Option

The number of asynchronous requests can be specified using the ICE command line option -Z. This is available as part of the LDAP destination handler.

To set the number of asynchronous requests sent by the ICE client to 50, you would enter the following command:

```
ice -SLDIF -f LDIF_file -a -c -DLDAP -d cn_of_admin -Z50 -w password
```

Increased Number of LDAP Writer Threads

The LDAP server now has multiple writer threads. Use the -F ICE command line option for enabling forward referencing to avoid any possible errors due to concurrent processing as follows:

```
ice -SLDIF -f LDIF_file -a -c -DLDAP -d cn_of_admin -w password  -F
```

Disabling Schema Validation in ICE

Use the -C and -n ICE command line options to disable schema validation at the ICE client as follows:

```
ice -C -n -SLDIF -f LDIF_file -a -c -DLDAP -d cn_of_admin -w password
```

Backlinker

Backlinker is a background process that checks the referential integrity among other checks runs 50 minutes after the OpenText eDirectory server comes up. The subsequent time it runs is after 13 hours. Ensure that backlinker does not run during the bulkload process. In case backlinker runs, depending on the time and the number of objects loaded, backlinker can hinder the bulkload

Disabling ACL Templates

You can disable the Access Control List (ACL) templates to increase the bulkload performance. The implication of this is that some of the ACLs will be missing. However, you can resolve this by adding the required ACLs to the LDIF file or applying them later.

1 Run the following command:

```
ldapsearch -D cn_of_admin -w password -b cn=schema -s base
objectclasses=inetorgperson
```

The output of this command would be similar to the following:

```
dn: cn=schema
objectClasses: ( 2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' SUP
 organizationalPerson STRUCTURAL MAY ( groupMembership $
ndsHomeDirectory
 $ loginAllowedTi meMap $ loginDisabled $ loginExpirationTime $
 loginGraceLimit $ loginGraceRem aining $ loginIntruderAddress $
 loginIntruderAttempts $ loginIntruderResetTim e $
 loginMaximumSimultaneous $ loginScript $ loginTime $
 networkAddressRestri ction $ networkAddress $ passwordsUsed $
 passwordAllowChange $ passwordExpirationInterval $
 passwordExpirationTime $passwordMinimumLength $ passwordRequired $
 passwordUniqueRequired $ printJobConfiguration $ privateKey $ Profile
$
 publicKey $ securityEquals $ accountBalance $ allowUnlimitedCredit $
 minimum AccountBalance $ messageServer $ Language $ UID $
 lockedByIntruder $ serverHolds $ lastLoginTime $ typeCreatorMap $
 higherPrivileges $ printerControl $ securityFlags $ profileMembership
$
 Timezone $ sASServiceDN $ sASSecretStore $ sASSecretStoreKey $
 sASSecretStoreData $ sASPKIStoreKeys $ userCertificate
 $nDSPKIUserCertificateInfo $ nDSPKIKeystore $ rADIUSActiveConnections
$
 rADIUS AttributeLists $ rADIUSConcurrentLimit $
rADIUSConnectionHistory
 $ rADIUSDefa ultProfile $ rADIUSDialAccessGroup $
rADIUSEnableDialAccess
```

```
 $ rADIUSPassword $ rADIUSServiceList $ audio $ businessCategory $
 carLicense $ departmentNumbe r $ employeeNumber $ employeeType $
 givenName $ homePhone $ homePostalAddress  $ initials $ jpegPhoto $
 labeledUri $ mail $ manager $ mobile $ pager $ ldap Photo $
 preferredLanguage $ roomNumber $ secretary $ uid $ userSMIMECertifica
te
 $ x500UniqueIdentifier $ displayName $ userPKCS12 ) X-NDS_NAME 'User'
X
-NDS_NOT_CONTAINER '1' X-NDS_NONREMOVABLE '1' X-NDS_ACL_TEMPLATES (
'2#subtree#[Self]#[All Attributes Rights]' '6#entry#[Self]#loginScript'
'1#subtree#[Root Template]#[Entry Rights]'
'2#entry#[Public]#messageServer' '2#entry#[Root
Template]#groupMembership' '6#entry#[Self]#printJobConfiguration'
'2#entry#[Root  Template]#networkAddress') )
```

2 In the output noted in the previous step, delete the information marked in bold.

3 Save the revised output as an LDIF file.

4 Add the following information to the newly saved LDIF file:

```
dn: cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( 2.16.840.1.113730.3.2.2 )-add:objectclasses
```

Therefore, your LDIF should now be similar to the following:

```
dn: cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( 2.16.840.1.113730.3.2.2)
-
add:objectclasses
objectClasses: ( 2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' SUP
 organization alPerson STRUCTURAL MAY ( groupMembership $
ndsHomeDirectory
 $ loginAllowedTimeMap $ loginDisabled $ loginExpirationTime $
 loginGraceLimit $ loginGraceRem aining $ loginIntruderAddress $
 loginIntruderAttempts $ loginIntruderResetTime $
 loginMaximumSimultaneous $ loginScript $ loginTime $
 networkAddressRestri ction $ networkAddress $ passwordsUsed $
 passwordAllowChange $ passwordExpirationInterval $
 passwordExpirationTime $ passwordMinimumLength $ passwordRequired
 $passwordUniqueRequired $ printJobConfiguration $ privateKey $ Profile
$
 publicKey $ securityEquals $ accountBalance $ allowUnlimitedCredit $
 minimum AccountBalance $ messageServer $ Language $ UID $
 lockedByIntruder $ serverHolds $ lastLoginTime $ typeCreatorMap $
 higherPrivileges $ printerControl $ securityFlags $ profileMembership
$
 Timezone $ sASServiceDN $ sASSecretStore $ sASSecretStoreKey $
 sASSecretStoreData $ sASPKIStoreKeys $ userCertificate $
 nDSPKIUserCertificateInfo $ nDSPKIKeystore $ rADIUSActiveConnections $
```

```
 rADIUSAttributeLists $ rADIUSConcurrentLimit $ rADIUSConnectionHistory
$
 rADIUSDefa ultProfile $ rADIUSDialAccessGroup $ rADIUSEnableDialAccess
 $rADIUSPassword $ rADIUSServiceList $ audio $ businessCategory $
 carLicense
 $ departmentNumbe r $ employeeNumber $ employeeType $ givenName $
 homePhone $ homePostalAddress  $ initials $ jpegPhoto $ labeledUri $
mail
 $ manager $ mobile $ pager $ ldap Photo $ preferredLanguage $
roomNumber
 $ secretary $ uid $ userSMIMECertifica te $ x500UniqueIdentifier $
 displayName $ userPKCS12 ) X-NDS_NAME 'User' X-ND S_NOT_CONTAINER '1'
X
-NDS_NONREMOVABLE '1')
```

5 Enter the following command:

```
ldapmodify -D cn_of_admin -w password -f LDIF_file_name
```

For more information on working with ACLs, refer to the OpenText™ eDirectory CE 24.4 (v9.3) Tuning Guide.

Enabling/Disabling Inline Cache

You can enable or disable the Inline Change Cache for a server. You can disable Inline Change Cache only when Outbound Synchronization is disabled. Enabling Outbound Synchronization also enables Inline Change Cache.

Disabling Inline Change Cache marks the change cache as invalid for this replica and tags it with an invalid flag in **Agent Configuration > Partitions**. Enabling Inline Change Cache removes the invalid change cache flag when the change cache is rebuilt.

Increasing the LBURP Time Out Period

By default, the time out period for a client is 20 minutes (1200 seconds). But during bulkload, with the LBURP transaction size as high as 250, objects with large number of attributes with huge values for these attributes, and with LBURP concurrent processing enabled at the server, the server gets busy processing data pumped in by the ICE client without responding to the client in the stipulated time. This times out the ICE client

Therefore, we recommend you to increase the time out period. You can do this by exporting the environment variable LBURP_TIMEOUT with high values (in seconds).

For example, to export the LBURP_TIMEOUT variable with 1200 seconds, enter the following:
`export ICE_LBURP_TIMEOUT=1200`

Using ldif2dib for Bulkloading

You can specify the LDIF file containing the data to be imported and the path to the database files where data needs to be imported through the command line interface. Using ldif2dib to bulkload data requires the following steps:

1 Take a backup of the DIB.

For more information on the backup and restore process, see Chapter 15, “Backing Up and Restoring OpenText eDirectory,” on page 417.

2 Stop the OpenText eDirectory server.

3 To start bulkloading from the LDIF file, enter the following at the command prompt:

```
ldif2dib <LDIF File Name> [Options]
```

Where

- **LDIF File Name:** Specifies the name of LDIF file to bulkload.
- **Options:** These are optional and specify the different parameters that you can use for tuning this utility. The options supported by the ldif2dib utility are listed below:

For example, if you want to set the options for specifying batch mode, cache size and block cache percentage options, enter the following command:

```
ldif2dib 1MillionUsers.ldif -b/novell/log/logfile.txt -c314572800 -p90
```

TIP: You can temporarily suspend the bulkload by pressing the s/S key. The Escape key (Esc) can be used to stop the bulkload.

Multiple Instances

ldif2dib can be used to bulkload entries from LDIF files to a particular instance of OpenText eDirectory (DIB) by specifying the location of its nds.db file with the -n option. If the location of the nds.db file is not specified with the -n option and if there is a single instance of OpenText eDirectory configured on the system, ldif2dib automatically detects the location of its database files. However, if there are multiple instances, ldif2dib displays a menu listing all configured instances and allows you to choose an instance for bulkload.

For more information on the multiple instances of OpenText eDirectory, see Using ndsconfig to Configure Multiple Instances of OpenText eDirectory 9.3 in the OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide.

Tuning ldif2dib

This section contains information about the parameters that can be used to tune ldif2dib:

- “Tuning the Cache” on page 197
- “Transaction Size” on page 197
- “Index” on page 197
- “Block Cache Percent” on page 197
- “Check Point Interval” on page 197

Tuning the Cache

The database cache setting is one of the more significant settings that affects the OpenText eDirectory performance. If it is set too low, eDirectory operations slow down because information must be retrieved from the disk more often. If it is set too high, enough memory is not available for other processes to run and the whole system slows down. For more information on cache, see Modifying FLAIM Cache Settings in the OpenText eDirectory Tuning Guide.

Bulkload performance generally increases on increasing the cache size. However, no performance improvement has been observed by increasing the cache size beyond a value which is 3.8 times the size of the LDIF file.

Transaction Size

The transaction size defines the chunk size in terms of number of objects per transaction. When the transaction size is high, a small number of large chunk writes result and when it is low, a large number of small chunk writes result.

The bulkload performance increases with higher transaction sizes. A transaction size of zero results in a special case which allows unlimited objects per transaction. When the transaction size is zero, the performance is high because the commit is done at the end of the bulkload. However, we do not recommend you to set the transaction size to 0 for very large LDIF files (larger than one million objects). You can set the transaction size as high as 4000 for very large LDIF files.

Index

Although use of indexes leads to a higher search performance, it makes bulkload slower because indexes need to be updated for every object loaded to the DIB. This is especially true for substring indexes. Therefore when you are bulkloading large number of objects, you can suspend indexes to speed up the bulkload. The indexes are automatically resumed when OpenText eDirectory server is brought up. Use the -x option to disable indexes before loading entries using ldif2dib.

Block Cache Percent

If the sub-string indexes are enabled for attributes, it is recommended to set the block cache percent to 50%, and if the sub-string indexes are disabled for attributes, you can set the block cache percent to 90%.

Check Point Interval

Checkpoint interval is the time for which the database waits before it initiates the checkpoint background thread which brings the on-disk version of the database up to the same coherent state as the in-memory (cached) database. This check point thread flushes the dirty cache to the disk, followed by cleaning up the roll forward log. Since bulkload is temporarily suspended while check point thread runs, we recommend that you set the check point interval to a high value to achieve faster bulkloads.

Limitations

This section contains limitations of the ldif2dib utility:

- "Schema" on page 198
- "ACL Templates" on page 198
- "Options" on page 198
- "Simple Password LDIF" on page 198
- "Custom Classes" on page 199
- "Filtered Replicas" on page 199

Schema

- The LDIF file should mention all the object classes that an entry belongs to. An entry can belong to multiple object classes because of inheritance. For example, an entry of type inetOrgPerson should have following syntax in the LDIF file:

  ```
  objectclass: inetorgperson
  objectclass: organizationalPerson
  objectclass: person
  objectclass: top
  ```

- Currently, following syntaxes are not supported:

ACL Templates

ACLs that are specified in the ACL templates for an object class, are not automatically added for objects bulkloaded using ldif2dib.

Options

On Linux, if the -b option is used, the screen that displays statistics disappears after the bulkload is complete. The final statistics, however, are written to the log file for reference.

Simple Password LDIF

On Windows, while uploading LDIF having simple password, ldif2dib might fail if the NICI keys in system and Administrator folder are not in sync. To work around this issue, access the keys present in the `nici/system` folder as follows:

1. Go to the `C:\Windows\system32\novell\nici\folder`.
2. Backup the files present in the **Administrator** folder.
3. Get access to the system folder and its files by following the below mentioned steps:
   - **3a** Go to the **Security** tab in the Properties window of the system folder.
   - **3b** Select **Advanced Options** and go to **Owner** tab.
   - **3c** Select **Administrator**.

3d Go back to the **Security** tab and add **Administrator** to the list.

Repeat the similar steps to get read access to all the files present inside the system folder.

4 Overwrite the files in the **Administrator** folder with the ones in the system folder.

5 Once the upload is done, copy the backed up files to the **Administrator** folder.

6 Revert the Administrator's access to the system folder and also the files within the folder.

Custom Classes

Bulkloading an LDIF with a large number of container objects using ldif2dib can result in a memory build up leading to a -150 error being reported.

Filtered Replicas

OpenText eDirectory does not support bulkloading operations to filtered replicas.

Caveats

Behavior of ldif2dib is undefined in the following scenarios:

- “Duplicate Entries” on page 199
- “No Schema Checks” on page 199
- “Insufficient Space on Hard-Drive” on page 199
- “Forced Termination” on page 200
- “Terminal Resizing” on page 200

Duplicate Entries

Uploading LDIF files having duplicate entries or having entries already present in the DIB, without the -u option would cause the entry to be added more than once, leading to an inconsistent state of the DIB. So if you are not sure if entries are repeated in the LDIF or if they are present in DIB before the bulkload, use the -u option during bulkload.

No Schema Checks

ldif2dib does not perform any schema checks. As a result, you can add an attribute to an object even if the attribute does not belong to the schema of the object. This would leave the DIB in an inconsistent state. Use ldif2dib only when you are sure that the LDIF data does not need schema checks.

Insufficient Space on Hard-Drive

Behavior of ldif2dib is undefined when there is not enough space on the hard-drive for all the objects being loaded. You need to make sure that there is sufficient space for all the objects before starting the bulkload.

Forced Termination

Forcefully terminating the ldif2dib process can leave the DIB in an inconsistent state. Use the Escape key to gracefully exit the bulkload.

Terminal Resizing

Resizing the terminal during bulkload can distort the statistics displayed on the user interface. Terminal resizing should be avoided while bulkload is in progress.

LDIF Files

The OpenText Import Conversion Export utility lets you easily import LDIF files into and export LDIF files from eDirectory. For more information, see "OpenText Import Conversion Export Utility" in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide*.

In order for an LDIF import to work properly, you must start with an LDIF file that the OpenText Import Conversion Export utility can read and process. This section describes the LDIF file format and syntax and provides examples of correct LDIF files.

- "Understanding LDIF" on page 200
- "Debugging LDIF Files" on page 209
- "Using LDIF to Extend the Schema" on page 211
- "ldif2dib Limitations" on page 216

Understanding LDIF

LDIF is a widely used file format that describes directory information or modification operations that can be performed on a directory. LDIF is completely independent of the storage format used within any specific directory implementation, and is typically used to export directory information from and import data to LDAP servers.

LDIF is usually easy to generate. This makes it possible to use tools like awk or perl to move data from a proprietary format into an LDAP directory. You can also write scripts to generate test data in LDIF format.

LDIF File Format

OpenText Import Conversion Export imports require LDIF 1 formatted files. The following are the basic rules for an LDIF 1 file:

- The first non-comment line must be version: 1.
- A series of one or more records follows the version.
- Each record is composed of fields, one field per line.
- Lines are separated by either a new line or a carriage return/new line pair.
- Records are separated by one or more blank lines.

- There are two distinct types of LDIF records: content records and change records. An LDIF file can contain an unlimited number of records, but they all must be of the same type. You can't mix content records and change records in the same LDIF file.
- Any line beginning with the pound sign (#) is a comment and is ignored when processing the LDIF file.

LDIF Content Records

An LDIF content record represents the contents of an entire entry. The following is an example of an LDIF file with four content records:

```
 1  version: 1
 2  dn: c=US
 3  objectClass: top
 4  objectClass: country
 5
 6  dn: l=San Francisco, c=US
 7  objectClass: top
 8  objectClass: locality
 9  st: San Francisco
10
11 dn: ou=Artists, l=San Francisco, c=US
12 objectClass: top
13 objectClass: organizationalUnit
14 telephoneNumber: +1 415 555 0000
15
16 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
17 sn: Michaels
18 givenname: Peter
19 objectClass: top
20 objectClass: person
21 objectClass: organizationalPerson
22 objectClass: iNetOrgPerson
23 telephonenumber: +1 415 555 0001
24 mail: Peter.Michaels@aaa.com
25 userpassword: Peter123
26
```

This LDIF file is composed of the following parts:

Component	Description
Version Specifier	The first line of an LDIF file contains the version. Zero or more spaces are allowed between the colon and the version number, which is currently defined to be 1. If the version line is missing, any application processing the LDIF file is allowed to assume that the file is version 0. It's also possible that the LDIF file could be rejected as syntactically incorrect. OpenText utilities that process LDIF assume a file version of 0 when the version line is missing.

Component	Description
Distinguished Name Specifier	The first line of every content record (lines 2, 6, 11, and 16 in the example above) specifies the DN of the entry that it represents. The DN specifier must take one of the following two forms: ◆ dn: *safe_UTF-8_distinguished_name* ◆ dn:: *Base64_encoded_distinguished_name*
Line Delimiters	The line separator can be either a line feed or a carriage return/line feed pair. This resolves a common incompatibility between Linux and Solaris text files, which use a line feed as the line separator, and MS-DOS* and Windows text files, which use a carriage return/line feed pair as the line separator.
Record Delimiters	Blank lines (lines 5, 10, 15, and 26 in the example above) are used as record delimiters. Every record in an LDIF file including the last record must be terminated with a record delimiter (one or more blank lines). Although some implementations will silently accept an LDIF file without a terminating record delimiter, the LDIF specification requires it.
Attribute Value Specifier	All other lines in a content records are value specifiers. Value specifiers must take on one of the following three forms: ◆ Attribute description: *value* ◆ Attribute description:: *Base64_encoded_value* ◆ Attribute description: < *URL*

LDIF Change Records

LDIF change records contain modifications to be made to a directory. Any of the LDAP update operations (add, delete, modify, and modify DN) can be represented in an LDIF change record.

LDIF change records use the same format for the distinguished name specifier, attribute value specifier, and record delimiter as LDIF content records. (See "LDIF Content Records" on page 201 for more information.) The presence of a `changetype` field is what distinguishes an LDIF change record from an LDIF content record. A `changetype` field identifies the operation specified by the change record.

A `changetype` field can take one of the following five forms:

Form	Description
changetype: add	A keyword indicating that the change record specifies an LDAP add operation.
changetype: delete	A keyword indicating that the change record specifies an LDAP delete operation.
changetype: moddn	A keyword indicating that the change record specifies an LDAP modify DN operation if the LDIF processor is bound to the LDAP server as a version 3 client or a modify RDN operation if the LDIF processor is bound to the LDAP server as a version 2 client.
changetype: modrdn	A synonym for the moddn change type.
changetype: modify	A keyword indicating that the change record specifies an LDAP modify operation.

The Add Change Type

An add change record looks just like a content change record (see “LDIF Content Records” on page 201) with the addition of the changetype: add field immediately before any attribute value fields.

All records must be the same type. You can’t mix content records and change records.

```
 1 version: 1
 2 dn: c=US
 3 changetype: add
 4 objectClass: top
 5 objectClass: country
 6
 7 dn: l=San Francisco, c=US
 8 changetype: add
 9 objectClass: top
10 objectClass: locality
11 st: San Francisco
12
14 dn: ou=Artists, l=San Francisco, c=US
15   changetype: add
16 objectClass: top
17 objectClass: organizationalUnit
```

```
18 telephoneNumber: +1 415 555 0000
19
20 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
21 changetype: add
22 sn: Michaels
23 givenname: Peter
24 objectClass: top
25 objectClass: person
26 objectClass: organizationalPerson
27 objectClass: iNetOrgPerson
28 telephonenumber: +1 415 555 0001
29 mail: Peter.Michaels@aaa.com
30 userpassword: Peter123
31
```

The Delete Change Type

Because a delete change record specifies the deletion of an entry, the only fields required for a delete change record are the distinguished name specifier and a delete change type.

The following is an example of an LDIF file used to delete the four entries created by the LDIF file shown in "The Add Change Type" on page 203.

IMPORTANT: To delete entries you have previously added, reverse the order of the entries. If you do not do this, the delete operation fails because the container entries are not empty.

```
 1 version: 1
 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 3 changetype: delete
 4
 5 dn: ou=Artists, l=San Francisco, c=US
 8   changetype: delete
 9
10 dn: l=San Francisco, c=US
11 changetype: delete
12
13 dn: c=US
14 changetype: delete
15
```

The Modify Change Type

The modify change type lets you to specify the addition, deletion, and replacement of attribute values for an entry that already exists. Modifications take one of the following three forms:

Element	Description
add: attribute type	A keyword indicating that subsequent attribute value specifiers for the attribute type should be added to the entry.

Element	Description
delete: attribute type	A keyword indicating that values of the attribute type are to be deleted. If attribute value specifiers follow the delete field, the values given are deleted. If no attribute value specifiers follow the delete field, then all values are deleted. If the attribute has no values, this operation will fail, but the desired effect will still be achieved because the attribute had no values to be deleted.
replace: attribute type	A keyword indicating that the values of the attribute type are to be replaced. Any attribute value specifiers that follow the replace field become the new values for the attribute type. If no attribute value specifiers follow the replace field, the current set of values is replaced with an empty set of values (which causes the attribute to be removed). Unlike the delete modification specifier, if the attribute has no values, the replace will still succeed. The net effect in both cases is the same.

The following is an example of a modify change type that will add an additional telephone number to the cn=Peter Michaels entry.

```
 1 version: 1
 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 3 changetype: modify
 4 # add the telephone number to cn=Peter Michaels
 4 add: telephonenumber
 5 telephonenumber: +1 415 555 0002
 6
```

Just as you can combine a mixture of modifications in a single LDAP modify request, you can specify multiple modifications in a single LDIF record. A line containing only the hyphen (-) character is used to mark the end of the attribute value specifications for each modification specifier.

The following example LDIF file contains a mixture of modifications:

```
 1 version: 1
 2
 3 # An empty line to demonstrate that one or more
 4 # line separators between the version identifier
 5 # and the first record is legal.
 6
 7 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 8 changetype: modify
 9 # Add an additional telephone number value.
10 add: telephonenumber
11 telephonenumber: +1 415 555 0002
12 -
13 # Delete the entire fascimiletelephonenumber attribute.
14 delete: facsimileTelephoneNumber
15 -
16 # Replace the existing description (if any exists)
```

```
17 # with two new values.
18 replace: description
19 description: guitar player
20 description: solo performer
21 -
22 # Delete a specific value from the telephonenumber
23 # attribute.
24 delete: telephonenumber
25 telephonenumber: +1 415 555 0001
26 -
27 # Replace the existing title attribute with an empty
28 # set of values, thereby causing the title attribute to
29 # be removed.
30 replace: title
31 -
32
```

The Modify DN Change Type

The modify DN change type lets you rename an entry, move it, or both. This change type is composed of two required fields and one optional field.

Field	Description
newrdn (required)	Gives the new name for the entry that will be assigned while processing this record. The new RDN specifier must take of the following two forms: ◆ newrdn: *safe_UTF-8_relative_distinguished_name* ◆ newrdn:: *Base64_encoded_relative_ distinguished_name* The new RDN specifier is required in all LDIF records with a modify DN change type.
deleteoldrdn (required)	The delete old RDN specifier is a flag that indicates whether the old RDN should be replaced by the newrdn or if it should be kept. It takes one of the two following forms: ◆ deleteoldrdn: 0 Indicates that the old RDN value should be kept in the entry after it is renamed. ◆ deleteoldrdn: 1 Indicates that the old RDN value should be deleted when the entry is renamed.

Field	Description
newsuperior (optional)	The new superior specifier gives the name of the new parent that will be assigned to the entry while processing the modify DN record. The new superior specifier must take of the following two forms: ◆ newsuperior: *safe_UTF-8_distinguished_name* ◆ newsuperior:: *Base64_encoded_distinguished_ name* The new superior specifier is optional in LDIF records with a modify DN change type. It is only given in cases where you want to re-parent the entry.

The following is an example of a modify DN change type that shows how to rename an entry:

```
 1 version: 1
 2
 3 # Rename ou=Artists to ou=West Coast Artists, and leave
 4 # its old RDN value.
 5 dn: ou=Artists,l=San Francisco,c=US
 6 changetype: moddn
 7 newrdn: ou=West Coast Artists
 8 deleteoldrdn: 1
 9
```

The following is an example of a modify DN change type that shows how to move an entry:

```
 1 version: 1
 2
 3 # Move cn=Peter Michaels from
 4 # ou=Artists,l=San Francisco,c=US to
 5 # ou=Promotion,l=New York,c=US and delete the old RDN.
 5 dn: cn=Peter Michaels,ou=Artists,l=San Francisco,c=US
 6 changetype: moddn
 7 newrdn: cn=Peter Michaels
 8 deleteoldrdn: 1
 9 newsuperior: ou=Promotion,l=New York,c=US
10
```

The following is an example of a modify DN change type that shows how to move an entry and rename it at the same time:

```
 1 version: 1
 2
 3 # Move ou=Promotion from l=New York,c=US to
 4 # l=San Francisco,c=US and rename it to
 5 # ou=National Promotion.
 5 dn: ou=Promotion,l=New York,c=US
 6 changetype: moddn
 7 newrdn: ou=National Promotion
 8 deleteoldrdn: 1
 9 newsuperior: l=San Francisco,c=US
10
```

IMPORTANT: The LDAP 2 modify RDN operation doesn't support moving entries. If you try to move an entry using the LDIF `newsuperior` syntax with an LDAP 2 client, the request will fail.

Line Folding within LDIF Files

To fold a line in an LDIF file, simply insert a line separator (a new line or a carriage return/new line pair) followed by a space at the place where you want the line folded. When the LDIF parser encounters a space at a beginning of the line, it knows to concatenate the rest of the data on the line with the data on the previous line. The leading space is then discarded.

You should not fold lines in the middle of a multibyte UTF-8 character.

The following is an example of an LDIF file with a folded line (see lines 13 and 14):

```
 1 version: 1
 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US
 3 sn: Michaels
 4 givenname: Peter
 5 objectClass: top
 6 objectClass: person
 7 objectClass: organizationalPerson
 8 objectClass: iNetOrgPerson
 9 telephonenumber: +1 415 555 0001
10 mail: Peter.Michaels@aaa.com
11 userpassword: Peter123
12 description: Peter is one of the most popular music
13  ians recording on our label. He's a big concert dr
14  aw, and his fans adore him.
15
```

Hashed Password Representation in LDIF Files

The hashed password is represented as base64 data in the LDIF file. The attribute name `userpassword` should be followed with the name of the encryption used for hashing the password. This name should be given within a pair of flower brackets "{ }" as shown below:

Example 1

For SHA hashed passwords:

```
1 version: 1 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US 3
sn: Michaels 4 userpassword: {SHA}xcbdh46ngh37jsd0naSFDedjAS30dm5
objectclass: inetOrgPerson
```

Example 2

For SHA hashed passwords:

```
1 version: 1 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US 3
sn: Michaels 4 userpassword: {SHA}sGs948DFGkakdfkasdDF34DF4dS3skl5DFS5
objectclass: inetOrgPerson
```

Example 3

For Digest MD5 hashed passwords:

```
1 version: 1 2 dn: cn=Peter Michaels, ou=Artists, l=San Francisco, c=US 3
sn: Michaels 4 userpassword: {MD5}a45lkSDF234SDFG62dsfsf2DG2QEvgdmnk4305
objectclass: inetOrgPerson
```

Debugging LDIF Files

- "Enabling Forward References" on page 209
- "Checking the Syntax of LDIF Files" on page 210
- "Using the LDIF Error File" on page 210
- "Using LDAP SDK Debugging Flags" on page 210

If you have problems with an LDIF file, consider the following:

Enabling Forward References

You might occasionally encounter LDIF files in which a record to add one entry comes before a record to add its parents. When this happens, an error is generated because the new entry's parent does not exist when the LDAP server attempts to add the entry.

To solve this problem, simply enable the use of forward references. When you enable the creation of forward references and an entry is going to be created before its parent exists, a placeholder called a forward reference is created for the entry's parent to allow the entry to be successfully created. If a later operation creates the parent, the forward reference is changed into a normal entry.

It is possible that one or more forward references will remain after your LDIF import is complete (if, for example, the LDIF file never created the parent for an entry). In this case, the forward reference will appear as an Unknown object in OpenText Identity Console. Although you can search on a forward reference entry, you cannot read attributes (except objectClass) from the forward reference entry because it does not have any attributes or attribute values. However, all LDAP operations will work normally on the real object entries located below the forward reference.

Identifying Forward Reference Entries

Forward reference entries have an object class of Unknown and also have their internal NDS EF_REFERENCE entry flag set. You can use LDAP to search for objects with an Unknown object class, although there is currently no way to access the entry flag settings through LDAP to be sure that they are forward reference entries.

Changing Forward Reference Entries into Normal Objects

You can change a forward reference entry into an normal object by simply creating it (using, for example, an LDIF file or an LDAP client request). When you ask OpenText eDirectory to create an entry that already exists as a forward reference, OpenText eDirectory transforms the existing forward reference entry into the object you asked it to create.

Using the OpenText Import Conversion Export Utility Command Line Interface

To enable forward references in the command line interface, use the -F LDAP destination handler option.

For more information, see "LDIF Destination Handler Options" in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide*.

Checking the Syntax of LDIF Files

You can check the syntax of an LDIF file before you process the records in the file by using the Display Operations But Do Not Perform LDIF source handler option.

The LDIF source handler always checks the syntax of the records in an LDIF file as it processes them. Using this option disables the processing of the records and lets you verify the syntax.

Using the OpenText Import Conversion Export Utility Command Line Interface

To check the syntax of an LDIF file in the command line interface, use the -n LDIF source handler option.

For more information, see "LDIF Source Handler Options" in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide*.

Using the LDIF Error File

The OpenText Import Conversion Export utility automatically creates an LDIF file listing any records that failed processing by the destination handler. You can edit the LDIF error file generated by the utility, fix the errors, then reapply it to the server to finish an import or data migration that contained failed records.

Using the OpenText Import Conversion Export Utility Command Line Interface

To configure error log options in the command line utility, use the `-l` general option.

For more information, see "General Options" in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide*.

Using LDAP SDK Debugging Flags

To understand some LDIF problems, you might need to see how the LDAP client SDK is functioning. You can set the following debugging flags for the LDAP source handler, the LDAP destination handler, or both.

Value	Description
0x0001	Trace LDAP function calls.
0x0002	Print information about packets.
0x0004	Print information about arguments.
0x0008	Print connections information.

Value	Description
0x0010	Print BER encoding and decoding information.
0x0020	Print search filter information.
0x0040	Print configuration information.
0x0080	Print ACL information.
0x0100	Print statistical information.
0x0200	Print additional statistical information.
0x0400	Print shell information.
0x0800	Print parsing information.
0xFFFF (-1 Decimal)	Enable all debugging options.

To enable this functionality, use the `-e` option for the LDAP source and LDAP destination handlers. The integer value you give for the `-e` option is a bitmask that enables various types of debugging information in the LDAP SDK.

For more information, see“LDAP Source Handler Options” and “LDAP Destination Handler Options” in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide*.

Using LDIF to Extend the Schema

Because LDIF can represent LDAP update operations, you can use LDIF to modify the schema.

Adding a New Object Class

To add a class, simply add an attribute value that conforms to the specification for `NDSObjectClassDescription` to the `objectClasses` attribute of the `subschemaSubentry`.

```
NDSObjectClassDescription = "(" whsp
   numericoid whsp
   [ "NAME" qdescrs ]
   [ "DESC" qdstring ]
   [ "OBSOLETE" whsp ]
   [ "SUP" oids ]
   [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
   [ "MUST" oids ]
   [ "MAY" oids ]
   [ "X-NDS_NOT_CONTAINER" qdstrings ]
   [ "X-NDS_NONREMOVABLE" qdstrings ]
   [ "X-NDS_CONTAINMENT" qdstrings ]
   [ "X-NDS_NAMING" qdstrings ]
   [ "X-NDS_NAME" qdstrings ]
   whsp ")"
```

The following example LDIF file adds the `person` objectClass to the schema:

```
 1 version: 1
 2 dn: cn=schema
 3 changetype: add
 4 objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard
 5   ObjectClass' SUP ndsLoginProperties STRUCTURAL MUST
 6   (cn $ sn) MAY (description $ seeAlso $ telephoneNum
 7  ber $ fullName $ givenName $ initials $ uid $ userPa
 8  ssword) X-NDS_NAMING ('cn' 'uid') X-NDS_CONTAINMENT
 9  ('organization' 'organizationalUnit' 'domain') X-NDS
10  _NAME 'Person' X-NDS_NOT_CONTAINER '1' X-NDS_NONREMO
11  VABLE '1')
12
```

Mandatory Attributes

Mandatory attributes are listed in the MUST section of the object class description. For the person object class, the mandatory attributes are cn and sn.

Optional Attributes

Optional attributes are listed in the MAY section of the object class description. The optional attributes in the person object class are description, seeAlso, telephoneNumber, fullName, givenName, initials, uid, and userPassword.

NOTE: The userPassword attribute cannot be used as an optional (MAY) attribute. The operation will fail if you try to use it as a mandatory (MUST) attribute in the new objectClass using this LDIF format to extend the schema.

Containment Rules

The object classes that can contain the object class being defined are given in the X-NDS_CONTAINMENT section of the object class description. The person object class can be contained by the organization, organizationalUnit, and domain object classes.

Adding a New Attribute

To add an attribute, simply add an attribute value that conforms to the specification for NDSAttributeTypeDescription to the attributes attribute of the subschemaSubentry.

```
NDSAttributeTypeDescription = "(" whsp
  numericoid whsp  ; AttributeType identifier
  [ "NAME" qdescrs ]  ; name used in AttributeType
  [ "DESC" qdstring ]  ; description
  [ "OBSOLETE" whsp ]
  [ "SUP" woid ]  ; derived from this other AttributeType
  [ "EQUALITY" woid]  ; Matching Rule name
  [ "ORDERING" woid]  ; Matching Rule name
  [ "SUBSTR" woid ]  ; Matching Rule name
  [ "SYNTAX" whsp noidlen whsp ] ;  Syntax OID
  [ "SINGLE-VALUE" whsp ]  ; default multi-valued
  [ "COLLECTIVE" whsp ]  ; default not collective
  [ "NO-USER-MODIFICATION" whsp ] ;  default user modifiable
  [ "USAGE" whsp AttributeUsage ] ;  default userApplications
```

```
 [ "X-NDS_PUBLIC_READ" qdstrings ]
                             ; default not public read ('0')
 [ "X-NDS_SERVER_READ" qdstrings ]
                              ; default not server read ('0')
 [ "X-NDS_NEVER_SYNC" qdstrings ]
                               ; default not never sync ('0')
 [ "X-NDS_NOT_SCHED_SYNC_IMMEDIATE" qdstrings ]
                            ;  default sched sync immediate ('0')
 [ "X-NDS_SCHED_SYNC_NEVER" qdstrings ]
                                 ;  default schedule sync ('0')
 [ "X-NDS_LOWER_BOUND" qdstrings ]
                              ;  default no lower bound('0')
                                   ;(upper is specified in SYNTAX)
 [ "X-NDS_NAME_VALUE_ACCESS" qdstrings ]
                         ; default not name value access ('0')
 [ "X-NDS_NAME" qdstrings ]  ; legacy NDS name
 whsp ")"
```

The following example LDIF file adds the `title` attribute type to the schema:

```
1 version: 1
2 dn: cn=schema
3 changetype: add
4 attributeTypes: ( 2.5.4.12 NAME 'title' DESC 'Standa
5  rd Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{
6  64} X-NDS_NAME 'Title' X-NDS_NOT_SCHED_SYNC_IMMEDIA
7  TE '1' X-NDS_LOWER_BOUND '1')
8
```

Single-Valued versus Multivalued

An attribute defaults to multivalued unless it is explicitly made single-valued. The following example LDIF file makes title single-valued by adding the `SINGLE-VALUE` keyword after the `SYNTAX` section:

```
1 version: 1
2 dn: cn=schema
3 changetype: add
4 attributeTypes: ( 2.5.4.12 NAME 'title' DESC 'Standa
5  rd Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{
6  64} SINGLE-VALUE X-NDS_NAME 'Title' X-NDS_NOT_SCHED
7  _SYNC_IMMEDIATE '1' X-NDS_LOWER_BOUND '1')
8
```

Adding an Optional Attribute to an Existing Object Class

Although adding new schema elements is an acceptable practice, modifying or extending existing schema elements is usually dangerous. Because every schema element is uniquely identified by an OID, when you extend a standard schema element, you effectively create a second definition for the element even though it still uses the original OID. This can cause incompatibility problems.

There are times when it is appropriate to change schema elements. For example, you might need to extend or modify new schema elements as you refine them during development. Instead of adding new attributes directly to a class, you should generally use auxiliary classes only to

- Add new attributes to an existing object class.
- Subclass an existing object class.

Adding or Removing Auxiliary Classes

The following sample LDIF file creates two new attributes, creates an auxiliary class with these new attributes, then adds an `inetOrgPerson` entry with the `auxiliary` class as an object class of the entry and with values for the `auxiliary` class attributes.

```
version: 1
# Add an attribute to track a bear's hair. The attribute is
# multi-valued, uses a case ignore string syntax,
# and has public read rights
# Values may include: long hair, short, curly, straight,
# none, black, and brown
# X-NDS_PUBLIC_READ '1' The 1 allows public read,
# 0 denies public read
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.10 NAME
'bearHair' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-NDS_PUBLIC_READ '1' )

# add an attribute to store a bear's picture
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.11 NAME
'bearPicture' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
SINGLE-VALUE )

# create an Auxiliary class for the bearfeatures
dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (2.16.840.1.113719.1.186.6.101 NAME
'bearFeatures' MAY (bearHair $ bearPicture) AUXILIARY)

# now create a user named bobby
dn: cn=bobby,o=bearcave
changetype: add
cn: bobby
```

```
sn: bear
givenName: bobby
bearHair: Short
bearHair: Brown
bearHair: Curly
bearPicture:< file:///c:/tmp/alien.jpg
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: bearFeatures

# now create a person named john that will later be changed
# into a bear when bearFeatures is added to its objectClass
# list
dn: cn=john,o=bearcave
changetype: add
cn: John
sn: bear
givenName: john
objectClass: top
objectClass: person
objectClass: inetOrgPerson

# now morph john into a bear by adding bearFeatures
dn: cn=john,o=bearcave
changetype: modify
add: objectClass
objectClass: bearFeatures
-
add: bearHair
bearHair: long
bearHair: black
#bearPicture:< file:///c:/tmp/john.jpg>
-

# to morph john back to a person, simply delete the
# objectClass bearFeatures
dn: cn=john,o=bearcave
changetype: modify
delete: objectClass
objectClass: bearFeatures
```

When removing auxiliary classes, you don't have to delete all of the values associated with the `auxiliary` class when you remove the `auxiliary` class from the objectClass list. OpenText eDirectory does this automatically.

If the `auxiliary` class had `MUST` attributes, they must all be specified in the same modify operation that adds the `auxiliary` class to the objectClass list, or the modification will fail.

Known Problems with XML Parsing

XML processing of any LDIF Record (LDIF format or records generated from LDAP server) will not succeed if the individual records will not satisfy all the XML rules specified in the XML file.

ldif2dib Limitations

- "Simple Password LDIF" on page 216
- "Schema" on page 216
- "ACL Templates" on page 216
- "Signal Handler" on page 217

Simple Password LDIF

On Windows, while uploading LDIF with a simple password, ldif2dib might fail if the NICI keys in the `system` and *`Administrator`* folders are not in sync.

To work around this issue, use the following procedure to access the keys in the `nici/system` folder:

1. Go to the `C:\Windows\system32\novell\nici\` folder (for 32-bit NICI).

   or

   Go to the `C:\Windows\SysWOW64\novell\nici\` folder (for 64-bit NICI).
2. Back up the files in the *`Administrator`* folder.
3. Go to the **Security** tab in the Properties window of the `system` folder.
4. Select **Advanced Options** and go to the **Owner** tab.
5. Select **Administrator**.
6. Go back to the **Security** tab and add Administrator to the list.
7. Repeat Step 3 through Step 6 to get read access to all the files inside the system folder.
8. Overwrite the files in the *`Administrator`* folder with the ones in the `system` folder.
9. After the upload is done, copy the backed-up files to the *`Administrator`* folder.
10. Change the Administrator's access to the `system` folder and also the files within the folder.

Schema

The LDIF file should mention all the object classes that an entry belongs to. You should also include the classes that an entry belongs to because of inheritance of classes. For example, an entry of type `inetOrgPerson` has following syntax in the LDIF file:

- `objectclass: inetorgperson`
- `objectclass: organizationalPerson`
- `objectclass: person`
- `objectclass: top`

ACL Templates

Objects bulkloaded using the `ldif2dib` utility are not added with ACLs that are specified in the ACL templates for the object class of the object.

Signal Handler

You can temporarily suspend the offline bulkload operation by pressing the s or S key. You can use the Escape key (Esc) to stop the bulkload operation.

8 Monitoring OpenText

OpenText provides cross-platform monitoring and diagnostic capability to all servers in your OpenText tree. This feature lets you monitor your servers from any location either by exposing multiple interfaces from different tools/protocol handlers or on your network where a Web browser is available. The OpenText utilities such as ndscheck, iMonitor, LDAP rootdse search, and ndsrepair help in gathering the monitoring data.

OpenText also provides an LDAP search method for retrieving the real time statistics for OpenText subsystems and background processes. In this method, OpenText records the state of OpenText processes and operations as an entry with the base DN of `cn=monitor`. By using this interface, an OpenText administrator can monitor the status of OpenText modules and operations. OpenText supports this feature on LDAP protocol and only an LDAP client can place requests for monitoring data.

- "Using OpenText iMonitor" on page 219
- "Using cn=monitor for Monitoring" on page 252
- "Using DSTrace" on page 263
- "DSTrace Messages" on page 271
- "OpenText iMonitor Message Filtering" on page 275
- "SAL Message Filtering" on page 275

Using OpenText iMonitor

OpenText iMonitor provides cross-platform monitoring and diagnostic capability to all servers in your OpenText tree. This utility lets you monitor your servers from any location on your network where a Web browser is available.

OpenText iMonitor lets you look at the OpenText environment in depth on a partition, replica, or server basis. You can also examine what tasks are taking place, when they are happening, what their results are, and how long they are taking.

OpenText iMonitor provides a Web-based alternative or replacement for many of the OpenText traditional server-based OpenText tools such as DSBrowse, DSTrace, DSDiag, and the diagnostic features available in DSRepair. Because of this, OpenText iMonitor's features are primarily server focused, meaning that they focus on the health of individual OpenText agents (running instances of the directory service) rather than the entire OpenText tree.

OpenText iMonitor provides the following features:

- OpenText eDirectory health summary
  - Synchronization information
  - Known servers
  - Agent configuration
- OpenText eDirectory health checks

- Hyperlinked DS Trace
- Agent configuration
- Agent activity and verb statistics
- Reports
- Agent information
- Error information
- Object/schema browser
- OpenText Identity Manager monitor
- Search
- Partition list
- Agent process status
- Background process schedule
- DSRepair
- Connection monitor

The information you can view in OpenText iMonitor is based the following factors:

- The identity you have established

  Your identity's OpenText eDirectory rights are applied to every request you make in OpenText iMonitor. For example, you must log in as the Administrator of the server or a console operator on the server where you are trying to access the DSRepair page.

- The OpenText eDirectory agent version you are monitoring

  Newer versions of NDS and OpenText eDirectory will have features and options that older versions do not.

The information you view in OpenText iMonitor immediately shows what is happening on your server.

This chapter gives information on the following topics:

- "System Requirements" on page 220
- "Accessing OpenText iMonitor" on page 221
- "OpenText iMonitor Architecture" on page 222
- "OpenText iMonitor Features" on page 227
- "Ensuring Secure OpenText iMonitor Operations" on page 249
- "Configuring HTTP Server Object" on page 250
- "Setting HTTP Stack Parameters Using ndsconfig" on page 251

System Requirements

To use OpenText iMonitor you need

- OpenText eDirectory 8.7.1 or later
- A supported Web browser, including Microsoft Internet Explorer or Firefox

Platforms

The OpenText iMonitor utility runs on the following platforms:

- Windows 2000 and 2003 Server (No SSL)
- Linux

For Windows, OpenText iMonitor loads automatically when OpenText eDirectory runs. On Linux, OpenText iMonitor can be loaded using the `ndsimonitor -l` command. It can also be loaded automatically by adding `[ndsimonitor]` in the `/etc/opt/novell/eDirectory/conf/ndsimon.conf` file before starting the OpenText eDirectory Server.

The OpenText iMonitor utility runs on the following Web browsers:

- Microsoft IE 10 & above
- Firefox* 40 & above

OpenText eDirectory Versions That Can Be Monitored

You can use OpenText iMonitor to monitor the following versions of NDS and OpenText eDirectory:

- All versions of NDS and OpenText eDirectory for Windows
- All versions of NDS and OpenText eDirectory for Linux

Accessing OpenText iMonitor

1. Ensure that the OpenText iMonitor executable is running on the OpenText eDirectory server.
2. Open your Web browser.
3. In the address (URL) field, enter

   `http://`*`server's_TCPIP_address`*`:`*`httpstack_port`*`/nds`

   for example:

   `http://137.65.135.150:8028/nds`

   DNS names can be used anywhere a server's IP or IPX address or distinguished name could be used in iMonitor. For example, when you have configured DNS, then

   `http://prv-gromit.provo.novell.com/nds?server=prv-igloo.provo.novell.com`

   is equivalent to

   `http://prv-gromit.provo.novell.com/nds?server=`*`IP_or_IPX address`*

   or

   `http://prv-gromit.provo.novell.com/nds?server=/cn=prv-igloo,ou=ds,ou=dev,o=novell,t=novell_inc`

   If an OpenText eDirectory HTTPS stack is available, you can use OpenText iMonitor through HTTPS.
4. Specify a user name, context, and password. For example, `login cn=admin.o=novell`

To have access to all of the features, log in as Administrator with the fully distinguished name, or as an administrator equivalent.

5 Click **Login**.

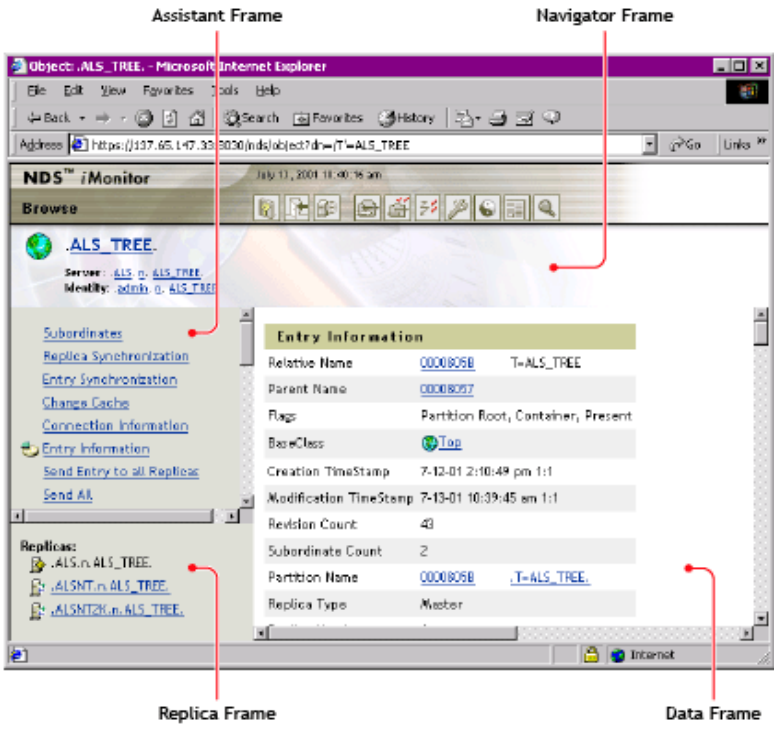
OpenText iMonitor Architecture

- "Anatomy of an OpenText iMonitor Page" on page 222
- "Modes of Operation" on page 223
- "iMonitor Features Available on Every Page" on page 224
- "Configuration Files" on page 225

Anatomy of an OpenText iMonitor Page

Each OpenText iMonitor page is divided into four frames or sections: the Navigator frame, the Assistant frame, the Data frame, and the Replica frame.

Figure 8-1 *OpenText iMonitor Frames*



Navigator Frame: Located across the top of the page. This frame shows the server name where the data is being read from, your identity, and the icons you can click to link to other screens, including online help, login, server portal, and other OpenText iMonitor pages.

Assistant Frame: Located at the left side of the page. This frame contains additional navigational aids, such as links to other pages, items that help you navigate data in the Data frame, or other items to assist you with obtaining or interpreting the data on a given page.

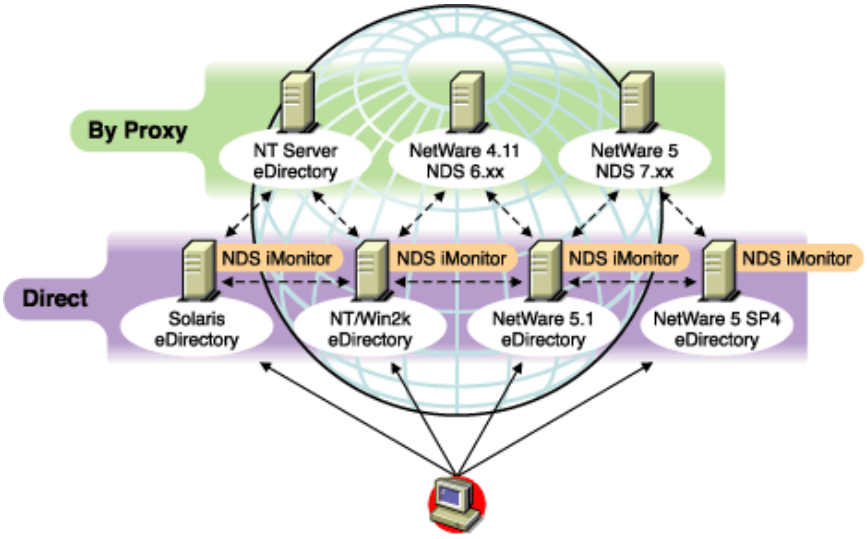
Data Frame: Shows the detailed information about your servers that you request by clicking one of the links listed above. This is the only page you will see if your Web browser does not support frames.

Replica Frame: Lets you determine which replica you are currently viewing and provides links to view the same information from another replica or server's point of view. This frame appears only when you view pages where another replica of the requested data exists or where another replica might have a different view of the information being presented in the Data frame.

Modes of Operation

OpenText iMonitor can be used in two different modes of operation: Direct mode and Proxy mode. No configuration changes are necessary to move between these modes. OpenText iMonitor automatically moves between these modes, but you should understand them in order to successfully and easily navigate the OpenText eDirectory tree.

Figure 8-2 *Modes of Operation*



Direct Mode: Use this mode when your Web browser is pointed directly at an address or DNS name on a machine running the iMonitor executable and reading information only on that machine's local eDirectory DIB.

Some OpenText iMonitor features are server-centric and are available only to the OpenText iMonitor running on that machine. These features use local API sets that cannot be accessed remotely. Server-centric features in OpenText iMonitor include the DSTrace, DSRepair, and Background Process Schedule pages. When using Direct mode, all OpenText iMonitor features will be available on that machine.

Key features of Direct mode:

- Full server-centric feature set
- Reduced network bandwidth (faster access)
- Access by proxy still available for all versions of OpenText eDirectory

Proxy Mode: Use this mode when your Web browser is pointed at an OpenText iMonitor running on one machine, but is gathering information from another machine. Because OpenText iMonitor uses traditional OpenText eDirectory non-server-centric protocols for non-server-centric features, all previous versions of OpenText eDirectory beginning with NDS 6.*x* can be monitored and diagnosed. However, server-centric features use APIs that cannot be accessed remotely.

If you are in Proxy mode and want to switch to Direct mode for a different server, you can do so as long as the server has a version of OpenText eDirectory in which OpenText iMonitor has shipped. If the server you are gathering information on by proxy has OpenText iMonitor running, you will see an additional icon button in the Navigator frame. When you move the mouse pointer over the icon, you will see a link to the remote OpenText iMonitor on the remote server. If the server you are gathering information on by proxy is an earlier version of OpenText eDirectory, no additional icon is shown and you will always need to gather information on that server by proxy until it is upgraded to a version of OpenText eDirectory that includes OpenText iMonitor.

Key features of Proxy mode:

- Not every server in the tree must be running OpenText iMonitor in order to use most OpenText iMonitor features
- Only one server must be upgraded
- There is a single point of access for dial-in
- You can access OpenText iMonitor over a slower speed link while OpenText iMonitor accesses OpenText eDirectory information over higher speed links
- Previous NDS version information is accessible
- Server-centric features are available only where iMonitor is installed

iMonitor Features Available on Every Page

You can link to the Agent Summary, Agent Information, Agent Configuration, Trace Configuration, DSRepair, Reports, and Search pages from any OpenText iMonitor page by using the icons in the Navigator frame. You can also log in or link to the OpenText Support Web page from any OpenText iMonitor page.

Login/Logout: The **Login** button is available if you are not logged in. A **Logout** button, which closes your browser window, is displayed if you are logged in. Unless all browser windows are closed, OpenText iMonitor session remains open, and you will not need to log in again. You can see your login status on any page by looking at Identity in the Navigator frame.

Support Connection Link: The OpenText logo in the upper right corner is a link to the OpenText Support Connection Web page. This provides a direct link to the OpenText Web site for current server patch kits, updates, and product-specific support.

Configuration Files

Configuration files are included with OpenText iMonitor to allow you to change or set default behavior or values in the utility.

The configuration files are text files containing configuration parameter tags together with their desired values. These files are located in the same directory as the OpenText iMonitor executable (which is usually in the same location as the OpenText eDirectory executables) on Windows, and in the `/etc` directory on Linux.

- "ndsimon" on page 225
- "ndsimonhealth" on page 226

ndsimon

The ndsimon configuration file lets you modify trace file settings, control access to the server, set the maximum number of object to be displayed when listing a container or displaying search results, and specify the number of minutes of inactivity allowed before a connection is logged out.

Server	Configuration File
Windows	*`install directory`*`\novell\NDS\ndsimon.ini`
Linux	`/etc/opt/novell/eDirectory/conf/ndsimon.conf`

There are two groups of parameters that you can set in the ndsimon configuration file.

- Parameters that apply to how the OpenText iMonitor executable itself runs

  When the OpenText iMonitor executable loads, it will attempt to listen on the traditional HTTP port 80. If that port is in use, it will back off to port 8028. If that port is in use, OpenText iMonitor will then back off again, increasing the port by 2 (8010, 8012, etc.) up to 8078.

  Where SSL is configured and available, a similar bind pattern is attempted. First, port 81 is tried, and then 8009, 8011, 8013, etc.

  This allows OpenText iMonitor to coexist with a Web server running on the same server. However, on some platforms, OpenText iMonitor might load before the installed Web server does, or you might want OpenText iMonitor to bind to a port of your choice. Both regular and SSL ports can be configured using the `HttpPort` and the `HttpsPort` parameters respectively.

- Parameters that apply to specific features or pages

  The configuration file that ships with OpenText iMonitor contains samples of the parameters that can be modified. These parameters are preceded by a pound sign (#). This indicates that they are commented out or not used when OpenText iMonitor parses the configuration file. For the shipping configuration file, OpenText iMonitor uses all internally bound default values for these parameters. To enable any of these parameters or to add any parameters, simply delete the # character from the beginning of the line.

ndsimonhealth

The ndsimonhealth configuration file lets you modify default settings for the Agent Health page. You can enable or disable Agent Health options, set reporting levels and ranges for options, and set server reporting levels.

Server	Configuration File
Windows	*install directory*\novell\NDS\ndsimonhealth.ini
Linux	/etc/opt/novell/eDirectory/conf/ndsimonhealth.conf

There are three types of options you can set in the ndsimonhealth configuration file.

- Enable/disable only options

  To disable an option, remove the pound sign (#) from in front of the option and replace any levels listed after the colon (:) with OFF. To set reporting levels of these options, remove the # character from in front of the option and add a reporting level after the colon. Valid levels are WARN, MARGINAL, and SUSPECT. For these options, you can input only one reporting level.

- General options that take a range of settings

  These options can be enabled and disabled or have their reporting level set, as well as the ranges for those reporting levels.

  To set the reporting level for any of these options, use the option name followed by -active: and the reporting levels you want. For example, to set time_delta active, add the following line to the configuration file:

  ```
  time_delta-active: WARN
  ```

  To set time_delta inactive, add the following line to the configuration file:

  ```
  time_delta-active: OFF
  ```

  When entering ranges, the specified range is the range that this reporting level should not be displayed for.

  See the time_delta example below for an example of how to set an option to be active for all three reporting levels and how to set the ranges. In this example, anything not in the range -2 to 2 is at least marginal, anything not in the range -5 to 5 is at least suspect, and anything not in the range -10 to 10 is a warning.

  ```
  time_delta-active: WARN | SUSPECT | MARGINAL
  time_delta-Min_Warn:      -10
  time_delta-Min_Suspect:    -5
  time_delta-Min_Marginal:   -2
  time_delta-Max_Marginal:    2
  time_delta-Max_Suspect:     5
  time_delta-Max_Warn:        10
  ```

  For help on any of these options, enter the following URL in iMonitor:

  ```
  http://XXX.XXX.XXX.XXX:PORT/nds/help?hbase=/nds/health/OPTION_NAME
  ```

XXX.XXX.XXX.XXX:*PORT* is the IP address and port where OpenText iMonitor can be reached, and *OPTION_NAME* is the name of the option you want help on (for example, time_delta).

To view the currently set levels and ranges, use your browser to go to the health page that contains the option you are interested in, then add the following to the end of the URL line in the browser:

```
&op=setup
```

- Options that need custom or complex settings

  There are three different server reporting levels that can be set:

  - WARN detects servers running a version of OpenText eDirectory that should be upgraded as soon as possible.
  - SUSPECT detects servers running a version of OpenText eDirectory that should be noted for upgrade.
  - MARGINAL detects servers running a version of OpenText eDirectory that is not current.

  These options set the reporting level if the server version falls within the specified range.

OpenText iMonitor Features

This section provides brief descriptions of OpenText iMonitor features.

Online help is provided in each section of OpenText iMonitor for more detailed information about each feature and function.

- "Viewing OpenText eDirectory Server Health" on page 228
- "Viewing Partition Synchronization Status" on page 228
- "Viewing Obituary Process Status and Change Cache Count" on page 229
- "Viewing Server Connection Information" on page 230
- "Viewing Known Servers" on page 231
- "Viewing Replica Information" on page 231
- "Controlling and Configuring the DS Agent" on page 232
- "Configuring Trace Settings" on page 233
- "Viewing Process Status Information" on page 234
- "Viewing Agent Activity" on page 234
- "Viewing Traffic Patterns" on page 235
- "Viewing Background Processes" on page 235
- "Configuring Background Processes" on page 235
- "Viewing OpenText eDirectory Server Errors" on page 236
- "Viewing DSRepair Information" on page 236
- "Viewing Agent Health Information" on page 236
- "Browsing Objects in Your Tree" on page 237
- "Viewing Entries for Synchronization or Purging" on page 237
- "Viewing OpenText Identity Manager Details" on page 238

- “Viewing the Synchronization Status of a Replica” on page 238
- “Configuring and Viewing Reports” on page 238
- “Viewing Schema, Class, and Attribute Definitions” on page 240
- “Searching for Objects” on page 241
- “Using the Stream Viewer” on page 241
- “Clone DIB Set” on page 242

Viewing OpenText eDirectory Server Health

From the Agent Summary page, you can view the health of your OpenText eDirectory servers, including synchronization information, agent process status, and the total servers known to your database.

1. In iMonitor, click **Agent Summary** .
2. Choose from the following options:

   Agent Synchronization Summary lets you view the number and types of replicas you have and the length of time since they have been successfully synchronized. You can also view the number of errors for each replica type. If there is only one replica or partition to view, the heading is **Partition Synchronization Status**.

   If the Agent Synchronization Summary doesn’t appear, there are no replicas you can view based on your identity.

   Servers Known to Database Totals lets you view the type and count of servers known to your database, and whether they are up or down.

   Agent Process Status Totals let you view the status of processes without the administrator's intervention that run on an agent. When there is a problem or piece of information, a status is recorded. The table increases or decreases, depending on the number of recorded statuses.

Viewing Partition Synchronization Status

From the Agent Synchronization page you can view the synchronization status of your partitions. You can filter the information by selecting from the options listed in the Assistant frame on the left side of the page.

1. In OpenText iMonitor, click **Agent Synchronization** in the Assistant frame.
2. Choose from the following options:

   Partition Synchronization Status lets you view the partition, number of errors, last successful synchronization, and maximum ring delta.

   Partition lets you view the links to each partition's Replica Synchronization page.

   Last Successful Sync lets you view the amount of time since all replicas of an individual partition were successfully able to synchronize from the server.

   Maximum Ring Delta shows the amount of data that might not be successfully synchronized to all the replicas in the ring. For example, if a user has changed his login script within the past 30 minutes, and the maximum ring delta has a 45-minute allocation, the user's login might not be

successfully synchronized, and he might get the previous login script when he attempts to log in. If, however, the user changed his login script more than 45 minutes ago, he should get the new login script consistently from all replicas.

If **Unknown** is listed under **Maximum Ring Delta**, it means the transitive synchronized vector is inconsistent and the maximum ring delta cannot be calculated due to replica/partition operations occurring, or some other problem.

Viewing Obituary Process Status and Change Cache Count

To view the obituary process status and the change cache count of a given partition, navigate to the partition root object of that partition. Data is displayed for three different types of obituaries:

- OBIT_DEAD: created when an object is deleted.
- OBIT_NEWRDN: created when an object is renamed.
- OBIT_MOVED: created when an object is moved from one location to another.

When the objects are processed, they can be in four different distinct states. They move from ISSUED state to PURGEABLE state, then finally get purged. Following are the four distinct states:

- ISSUED
- NOTIFIED
- OK_TO_PURGE
- PURGEABLE

There are 12 different distinct combinations for a given object. Following are the distinct combinations:

- OBIT_DEAD_ISSUED
- OBIT_DEAD_NOTIFIED
- OBIT_DEAD_OK_TO_PURGE
- OBIT_DEAD_PURGEABLE
- OBIT_NEWRDN_ISSUED
- OBIT_NEWRDN_NOTIFIED
- OBIT_NEWRDN_OK_TO_PURGE
- OBIT_NEWRDN_PURGEABLE
- OBIT_MOVED_ISSUED
- OBIT_MOVED_NOTIFIED
- OBIT_MOVED_OK_TO_PURGE
- OBIT_MOVED_PURGEABLE

A number is displayed against each of these combinations, which denotes the total number of objects that are in a particular state at the end of the last obituary processing cycle.

The change cache count displays the number of objects present in the change cache of the partition in the current server. The following figure shows the obit count and the change cache count for a particular partition root object of that partition.

Figure 8-3 *Obit and Change Cache Count Information*

Obit and Change Cache Count Information	
OBIT_DEAD_ISSUED	8318
OBIT_DEAD_NOTIFIED	0
OBIT_DEAD_OK_TO_PURGE	1682
OBIT_DEAD_PURGEABLE	0
OBIT_NEWRDN_ISSUED	0
OBIT_NEWRDN_NOTIFIED	0
OBIT_NEWRDN_OK_TO_PURGE	0
OBIT_NEWRDN_PURGEABLE	0
OBIT_MOVED_ISSUED	0
OBIT_MOVED_NOTIFIED	0
OBIT_MOVED_OK_TO_PURGE	0
OBIT_MOVED_PURGEABLE	0
Obit Count from database index	10000
Change Cache Count	10002

Viewing Server Connection Information

From the Agent Information page you can view the connection information for your server.

1. In OpenText iMonitor, click **Agent Information** in the Assistant frame.
2. Choose from the following options:

   Ping Info shows that OpenText iMonitor has attempted an IP ping to the set of addresses being advertised for the server. Success is as indicated.

   DNS Name shows that OpenText iMonitor has attempted to do an address reversal on IP addresses supported by the server and is indicating the associated DNS name.

   Depending on the transport, configuration, and platform you are running on, you might not see this information.

   Connection Information lets you view connection information for the server, including the server referral, time delta, Root Most Master, and replica depth.

   Depending on the transport, configuration, and platform you are running on, you might not see this information.

   Server Referral lets you view the set of addresses by which your server can be reached.

   Time Synchronized indicates that synthetic or future time is not being used unless a replica's last-issued time stamp is greater than the current time.

   OpenText eDirectory believes time is synchronized well enough to issue time stamps based on the server's current time. The time synchronization protocol might or might not currently be in a synchronized state.

Time Delta lets you view the difference in time between OpenText iMonitor and the remote server in seconds. A negative integer indicates that OpenText iMonitor's time is ahead of the server's time. A positive integer indicates that OpenText iMonitor's time is slower than the server.

Root Most Master specifies that the replica that is highest or closest to the root of the naming tree is a master replica.

Replica Depth lets you view the depth of the rootmost replica (the number of levels between the rootmost replica and the root of the tree).

Viewing Known Servers

From the **Known Servers** List, you can view the list of servers known to the database of the source server. You can filter the list to show all servers known to the database or to show all servers in the replica ring. If a server has an icon next to it, the server participates in a replica ring.

1. In OpenText iMonitor, click **Known Servers** in the Assistant frame.
2. Choose from the following options:

   Entry ID lists the identifier on the local server for an object. Entry IDs cannot be used across servers.

   NDS Revision lists the OpenText eDirectory build number or version being cached or stored on the server that you are communicating with.

   Status shows whether the server is up, down, or unknown. If the status shows as unknown, this means that this server has never needed to communicate with the server being shown as unknown.

   Last Updated shows the last time this server attempted to communicate with the server and found out it was down. If this column is not showing, all servers are currently up.

Viewing Replica Information

From the Partitions page, you can view information about the replicas on the server you are communicating with. You can filter the page by selecting from the options in the Assistant frame on the left side of the page.

Server Partition Information let you view information about the server's partition, including the entry ID, replica state, purge time, and last modification time.

Partition let you view information about the partition Tree object on the server.

Purge Time indicates the time when you can remove previously deleted data from the database because all replicas have seen the deletion.

Last Modification Time lets you view the last-issued time stamp of data written to the database for the replica. This lets you see if time is in the future and if synthetic time is being used.

Replica Synchronization lets you view the Replica Synchronization Summary page that refers to the partition. The Replica Synchronization page shows information about the partition synchronization status and replica status. You can also view lists of partitions and replicas.

Controlling and Configuring the DS Agent

From the Agent Configuration page, you can control and configure the DS Agent. The functionality you have on this page will depend on the rights of the current identity and the version of eDirectory you are looking at.

1. In OpenText iMonitor, click **Agent Configuration** .
2. Choose from the following options:
   - **Agent Information** let you view the connection information for your server.
   - **Partitions** lets you view the replicas on the server you are communicating with.
   - **Replication Filters** lets you view the replication filters configured for the specified OpenText eDirectory agent. NDS OpenText eDirectory 8.5 (build version 85.*xx*) was the first OpenText eDirectory version to implement a feature known as Filtered Replicas. See "Filtered Replicas" on page 60 for more information on what Filtered Replicas are, why they are used, and how to configure them.
   - **Agent Triggers** initiate certain background processes. These triggers are equivalent to using the SET DSTRACE=*`option` command.
   - **Background Process Settings** modify the interval at which certain background processes run. These settings are equivalent to the SET DSTRACE=!`option` command.
   - **Agent Synchronization** lets you disable or enable inbound or outbound synchronization. You can specify in hours the amount of time you want synchronization disabled.
   - **Database Cache** lets you configure the amount of database cache used by the DS database engine. Various cache statistics are also provided to assist you in determining whether you have an appropriate amount of cache available. Having an inadequate amount of cache might severely impact your system's performance.

     NOTE: We recommend you to allocate database cache after considering the file system cache and the available RAM.

   - **Login Settings** allows you you to specify whether OpenText eDirectory updates login attributes when users log in. The following options control how OpenText eDirectory responds when a user logs in:
     - **Login Update Delay** specifies the amount of time (in seconds) between updates. For example, if one or more users log in during the delay, OpenText eDirectory adds any changes to a queue. When the delay is over, OpenText eDirectory applies all queued changes.
     - **Login Update Disable Interval** specifies an interval of time (in seconds) during which the login attributes for a specific user will not be updated. A typical interval is 3600 seconds (1 hour). For example, when a user logs in for the first time at 8:00 AM, OpenText eDirectory updates attributes, and the interval starts. If the user logs in again before 9:00 AM, OpenTexteDirectory does not update the attributes. The default is 0, which means no disable interval is set.

Configuring Trace Settings

To access information on the Trace Configuration page, you must be the equivalent of Administrator of the server or a console operator. You are prompted to enter your user name and password so your credentials can be verified before you can access information on this page.

From the Trace Configuration page, you can set trace settings. OpenText iMonitor's DSTrace is a server-centric feature. That is, it can be initiated only on a server where iMonitor is running. If you need to access this feature on another server, you must switch to the iMonitor running on that server.

1. In iMonitor, click **Trace Configuration** .
2. Choose from the following options:
   - **Update** lets you submit changes to Trace Options and Trace Line Prefixes. If DSTrace is off, click **Trace On** to turn it on. If DSTrace is already on, click **Update** to submit changes to the current trace.
   - **Trace On/Off** turns DSTrace on or off. The button text changes based on the current DSTrace state. If DSTrace is on, the button text will read **Trace Off**. Clicking it toggles DSTrace between off and on. When DSTrace is off, clicking **Trace On** is equivalent to clicking **Update**.
   - **Trace Line Prefixes** lets you choose which pieces of data are added to the beginning of any trace line.
   - **DS Trace Options** apply to the events on the local DS Agent where the trace is initiated. The options show errors, potential problems, and other information about OpenText eDirectory on your local server. Turning on DS Trace options can increase CPU utilization and might reduce your system's performance. Therefore, DS Trace should generally be used for diagnostic purposes, not as a standard practice. These options are a more convenient equivalent of the SET DSTRACE=+*option* command.
   - **Event Configuration** lists the OpenText eDirectory and NMAS event options you can enable or disable for monitoring in DSTrace. The event system generates events for local activities such as adding objects, deleting objects, and modifying attribute values. For each type of event, a structure is returned that contains information specific to that type of event.
   - **Trace History** lets you view a list of previous trace runs. Each previous trace log is identified by the period of time during which the trace data was being gathered.
   - **Trace Triggers** let you view the trace flags that must be set in order to display the specified DS Agent information in DSTrace. These triggers might write large quantities of information to trace. Generally, we recommend that these triggers be enabled only when instructed by OpenText Support.
3. Click **Trace On** to turn DS Trace on and submit any changes.
4. Click  or **Trace Live** to view DS Trace in OpenText iMonitor.

Viewing Process Status Information

From the Agent Process Status page, you can view background process status errors and more information about each error that occurred. You can filter the information on this page by selecting from the options listed in the Assistant frame on the left side of the page.

In OpenText iMonitor, click **Agent Process Status** in the Assistant frame. Background process statuses that are currently reported include the following:

- Schema synchronization
- Obituary processing
- External reference/DRL
- Limber
- Repair

Viewing Agent Activity

From the Agent Activity page, you can determine traffic patterns and potential system bottlenecks. You can use this page to view the verbs and requests that are currently being handled by OpenText eDirectory. You can also see which of those requests are attempting to obtain DIB locks in order to write to the database and how many of those requests are waiting to obtain a DIB lock.

If you are viewing a server running OpenText eDirectory 8.6 or later, you will also see a list of partitions and the servers that participate in the replica ring with the server specified in the Navigator frame. With the introduction of OpenText eDirectory 8.6, synchronization is no longer single threaded. Any OpenText eDirectory 8.6 or later version server might outbound multiple partitions simultaneously to one or more replication partners. For this reason, the synchronization activity page was created so you can more easily monitor this parallel synchronization strategy.

1. In OpenText iMonitor, click **Agent Activity** in the Assistant frame.
2. Choose from the following options:
   - **Verb Activity and Statistics** lets you view a running count of all verbs called and requests made since OpenText eDirectory was last initialized. These pages also shows how many of those requests are currently active and the minimum, maximum, and average times (shown in milliseconds) that it takes to process those requests.
   - **Synchronization Current and Schedule** lists different times that inbound and outbound synchronization occurred. If inbound or outbound synchronization is currently taking place, you see an icon indicating that the process is active, when that cycle was started, and which server it is occurring with.

     If inbound and outbound synchronization is disabled, you see an icon indicating that fact and when it is scheduled to be re-enabled. For outbound synchronization, the next scheduled time is also shown.
   - **Events** lets you view a list of the currently active events, statistics for event handlers and a summary of event statistics, and the current event rights functions that have been called.
   - **Background Process Schedule** lets you view the background processes that are scheduled, what their current state is, and when they are scheduled to run again.

Viewing Traffic Patterns

From the Verb Statistics page, you can determine traffic patterns and potential system bottlenecks. You can use this page to view a running count of all verbs called and requests made since OpenText eDirectory was last initialized. This page also shows how many of those requests are currently active and the minimum, maximum, and average times (in milliseconds) it takes to process those requests. Background process, bindery, and standard OpenText eDirectory requests are tracked.

If you view this page on an older version of OpenText eDirectory, you might not see as much information as if you are running OpenText eDirectory 8.5 or later.

Viewing Background Processes

From the Background Process Schedule page, you can view the background processes that are scheduled, what their current state is, and when they are scheduled to run again. OpenText iMonitor's Background Process Schedule is a server-centric feature. That is, it can only be viewed on a server where OpenText iMonitor is running. If you need to access the background process schedule on another server, you must switch to the OpenText iMonitor running on that server. As you upgrade more servers to OpenText eDirectory 8.5 or later versions, OpenText iMonitor's server-centric features will be more available to you. Other server-centric features include the DSTrace and DSRepair pages.

To access information on the Background Process Schedule page, you must be the equivalent of Administrator of the server or a console operator. You are prompted to log in so your credentials can be verified before you can access information on this page.

Configuring Background Processes

To decrease how long background process cycles run, administrators can configure one of the following Background Process Delay Settings policies on the Background Process Settings window in OpenText iMonitor:

- CPU
- Hard Limit
- Purger Delay

To configure the background process:

1. Log into OpenText iMonitor.
2. Go to **Agent Configuration** > **Background process settings**.
3. Scroll down to the **Background Process Delay Settings** section and set the delay interval to any value from 0 through 100 milliseconds.

   By default, the **Hard Limit policy** is enabled with all the three processes sleeping for 100 milliseconds.

   or

   Select the **CPU Policy** and configure as appropriate.

   By default, the `Maximum CPU utilization %` parameter is set to 80% and `Maximum Delay Limit` is set to 100 milliseconds.

4 In the **Purger Interval** field, enter the delay interval.

By default, it is set to 30 minutes. You can change it depending on your requirement.

Viewing OpenText eDirectory Server Errors

From the Error Index page, you can view information about the errors found on your OpenText eDirectory servers. The errors are separated into two fields: OpenText eDirectory-specific errors and other errors that might be of interest. Each error listed is hyperlinked to a description that contains an explanation, possible cause, and troubleshooting actions.

1 In OpenText iMonitor, click **Error Index** in the Assistant frame.

From the Error Index page you can link to the latest OpenText documentation on errors, technical information, and white papers.

Viewing DSRepair Information

From the DSRepair page, you can view problems and back up or clean up your DIB sets. OpenText iMonitor's DSRepair is a server-centric feature. That is, it can be initiated only on a server where OpenText iMonitor is running. If you need to access the DSRepair information on another server, you must switch to the OpenText iMonitor running on that server. As you upgrade more servers to later versions of OpenText eDirectory, iMonitor's server-centric features will be more available to you. Other server-centric features include the DSTrace and Background Process Schedule pages.

To access information on this page, you must be the equivalent of Administrator of the server or a console operator. You are prompted to log in so your credentials can be verified before you can access information on this page.

1 In OpenText iMonitor, click **DSRepair** .

2 Choose from the following options:

- **Downloads** lets you retrieve repair-related files from the file server. You will not be able to access `dsrepair.log` if the DSRepair utility is running or you have initiated a repair from the DSRepair page in OpenText iMonitor until the operation is finished.
- **Delete Old DIB Sets** lets you delete an old DIB set by clicking the red **X**.

  WARNING: This action is irreversible. When you select this option, the old DIB set will be purged from the file system.

- **DS Repair Advanced Switches** lets you fix problems, check for problems, or create a backup of your database. You will not need to enter information in the **Support Options** field unless you are directed to do so by OpenText Support.

3 Click **Start Repair** to run DS Repair on this server.

Viewing Agent Health Information

From the Agent Health page, you can view health information about the specified OpenText eDirectory agent and the partitions and replica rings it participates in.

1 In OpenText iMonitor, click **Agent Health** in the Assistant frame.

2 Click the links to view detailed information.

Browsing Objects in Your Tree

From the Browse page, you can browse any object in your tree. The Navigation bar at the top of the page lets you know what server the object you are viewing is on, and the path to the object. The Replica frame on the left of the page lets you view or access the same object on any real partition. Click any underlined object on the page to view more information about an object. You can also click any portion of the name in the Navigator frame to browse up the tree.

The information displayed on this page depends on the OpenText eDirectory rights you are logged in with, the type of object you are browsing, and the version of NDS or OpenText eDirectory you are running. This page displays XRef objects if you are logged in with Supervisor rights. You can use the replica list to jump to a real copy of the replica. If you are browsing for objects in dynamic groups, the time stamp will not be displayed for the dynamic members.

Replica Synchronization displays the synchronization status of the replica that contains this object.

Entry Synchronization shows which attributes need to be synchronized from this server's point of view.

Connection Information indicates where OpenText iMonitor got the information for this object.

Entry Information displays the names, flags, base class, modification time stamp, and summary of connection information for the object.

Send Entry to All Replicas resends this entry's attributes to all other replicas. This process could take some time if the object has many attribute values. This does not make all other copies of the object identical. It simply allows the other replicas to reconsider each attribute.

Send All (visible only if the object being browsed is a partition root and the **Advanced Mode Option** is enabled) resends all entries in this partition to all the servers holding replicas of the partition. This does not make all copies of the objects being sent identical. It simply allows the other replicas to reconsider each object and its attributes.

Viewing Entries for Synchronization or Purging

From the Change Cache page, you can view a list of entries that this server needs to consider for synchronization or purging. This option is available only if the server you are accessing is running OpenText eDirectory 8.6 or later and the object you are viewing is a partition root. You must have Supervisor rights to the OpenText eDirectory server to view this page.

Entry Synchronization lets you determine why an entry needs to be synchronized.

NOTE: OpenText iMonitor only lists a limited number of objects in the Change Cache page. If you want to view all objects in the change cache, either for a specific partition or for all partitions on a server, you can run a Change Cache Dump Report in the Reports page. See "Configuring and Viewing Reports" on page 238 for more information about configuring and running reports in OpenText iMonitor.

Viewing OpenText Identity Manager Details

From the DirXML Summary page, you can view a list of any DirXML drivers running on your server, the status of each driver, any pending associations, and driver details.

1. In OpenText iMonitor, click **DirXML Summary** .
2. Choose from the following options:

   Status displays the current state of the specified driver. Possible states include stopped, starting, running, shut down, pending, and getting schema.

   Start Option displays the current startup option specified for the selected driver.

   Pending displays the number of associations that have not yet been made.

   Driver Details Icon displays subscriber and publisher details, XML rules, filters, and pending association lists for DirXML drivers running on your server. Details on the first 50 pending objects are also displayed on this page. The XML rule details provided on this page can be used to determine what to look for in the pending objects to allow their creation to proceed for the specified DirXML driver.

Viewing the Synchronization Status of a Replica

From the Replica Synchronization page, you can view the synchronization status of a replica.

1. In OpenText iMonitor, click **Agent Synchronization** in the Assistant frame.
2. Click **Replica Synchronization** for the partition you want to view.
3. Use the links on this page and in the navigation bar on the left to access other partitions and jump through your replica ring.

Configuring and Viewing Reports

From the Reports page, you can view and delete reports run directly on this server. Some reports might take a long time to run and can be resource intensive.

Scheduled reports run without authenticating as a user, using the [Public] identity. Any reports you run directly are run as your identity. All report data is stored on the server from which you run the report. iMonitor stores report data in the following directories by default, depending on the operating system:

Platform	Directory
Windows	`C:\Novell\NDS\ndsimon\dsreports\`
Linux	`/var/opt/novell/eDirectory/data/dsreports`

The Report Config page lets you view a list of preconfigured, custom, and scheduled reports. Use this page to modify and run reports and to create custom reports for OpenText iMonitor pages. The following table lists preconfigured reports included with OpenText iMonitor.

Report	Description
Server Information	Walks the entire tree, communicates with every NCP server it can find, and reports any errors it finds. Use this report to diagnose time synchronization and limber problems, or to find out if the current server is able to communicate with all other servers from this server's perspective. If selected in the Configuration page, this server can also generate NDS Agent Health information for every server in the tree.
Obituary Listing	Lists all obituaries on this server.
Object Statistics	Evaluates the objects in a given scope, then generates lists of objects matching the requested criteria. These criteria include such things as future time, unknown objects, renamed objects, counts of base classes, containers, alias, and external references.
Change Cache Dump	Lists all the objects in the change cache for the selected partition or for all partitions on the server. This report also generates an XML dump of the objects in the change cache, along with attributes and values that need to be synchronized across servers. The report provides information for analyzing all objects in the change cache. **NOTE:** OpenText iMonitor stores change cache dumps in the same directory as the actual Change Cache Dump Report, as listed in the previous table.
Service Advertising	Lists all directories and servers known to the current server through SLP or SAP.
Agent Health	Gathers health information for the current server.
Value Count	Generates a list of objects with attribute, which have value count more than a value you specify.

Viewing and Deleting Reports

1. In OpenText iMonitor, click **Reports** .
2. Click  to delete a report or  to view a report.

Running a Report

1. In OpenText iMonitor, click **Reports** > **Report Config**.
2. Click  to run a report.

Configuring or Scheduling a Report

1. In OpenText iMonitor, click **Reports** > **Report Config**.
2. Click  to configure and schedule a report.
3. Select any options you want, then click **Save Defaults** to save the options you selected.

4 (Optional) Configure the report to run either periodically or at a later time.

4a Specify a frequency, start time, and start day.

4b Click **Schedule**.

5 Click **Run Report** to start the report.

Creating a Custom Report

Custom reports let you launch any OpenText iMonitor page as a report.

1 In OpenText iMonitor, click **Reports** > **Report Config**.

2 In the **Runable Report** list, click  **Custom Reports**.

3 Enter a name for the report, then enter the URL for the OpenText iMonitor page you want to launch as a report.

When running a custom report, enter the URL as follows:

`/nds/required page`

4 In the **Saved reports** field, specify the number of versions of the report you want to keep or retain.

5 (Optional) Click **Save** to save the report.

6 (Optional) Configure the report to run either periodically or at a later time.

6a Specify a frequency, start time, and start day.

6b Click **Schedule**.

7 Click **Run Report** to start the report.

Viewing Schema, Class, and Attribute Definitions

From the Schema page, you can view your schema, class, and attribute definitions. You can view the schema that is loaded on your tree, with any extensions that have been made, and information specific to your particular schema, such as any changes or extensions you've made to the schema.

1 In OpenText iMonitor, click **Schema** in the Assistant frame.

2 Choose from the following options:

Synchronization List lists the servers that this server will synchronize with. This option is available only for servers running NDS OpenText eDirectory 8.5 or later. You must have Supervisor rights on the server to view this information.

Schema Root displays information about the schema replica closest to the root of the tee in this context.

Each OpenText eDirectory server stores a replica of the schema in its entirety. The schema replica is stored separately from the partitions that contain directory objects. Changes to any one schema replica are propagated to the other replicas. You can perform modifications to the schema only through a server that stores a writable replica of the root partition. Servers storing read-only replicas of the root partition can read but not modify schema information.

Attribute Definitions lists the name of each attribute, the syntax that the attribute value will be in, and the constraints that the attribute operates under. Use the navigation frame on the left to browse for and access individual attributes.

Class Definitions lists the name of each class, its rules, and its attributes. Use the navigation frame on the left to browse for and access individual attributes.

Searching for Objects

From the Search page, you can search objects based on a variety of query options and filters. The search query options and filters are grouped in two levels of search request forms: basic and advanced. The basic search request form is designed for average users of OpenText eDirectory and simple searches. The advanced search request form is designed for advanced users and complicated searches. Currently, only server-level search is supported.

All the search options and filters in the four sections are conjunctive. Blank fields (except the Relative Distinguished Name) will be ignored. Use the Ctrl key to deselect an item or select more than one item on the multilists. Deselected multilists will also be ignored.

1. In OpenText iMonitor, click **Search** .
2. Choose from the following options:
   - **Scope Options** lets you specify the scope of the search.
   - **Entry Filters** lets you specify search query filters related to the entry information.
   - **Attribute and Value Filters** lets you specify search query filters related to the attributes and values.
   - **Display Options** lets you specify options which control the display format of the search results.

     NOTE: The **Display Options** settings are only available if you click **Advanced** to view all Advanced Search options.
3. Click the **Help** button at the bottom of the search request form to see brief help information added to the form itself.

   Click **Reload** or **Refresh** to clear the help information.

Using the Stream Viewer

From the Stream Viewer page, you can view the current stream in any of the following formats:

- Plain text
- HTML
- GIF
- JPEG
- BMP
- WAV
- Hex Dump
- Other

If you have stream attributes that you consistently want to view in a particular format, you can use the Stream Viewer to select default display settings.

NDS Stream Attribute Setup changes the default display format for streams in your browser. It is up to your browser to display the stream correctly, so it might not always apply the settings you have selected.

You must be authenticated to the server to apply any changes you have made to the default settings. Your changes are stored in `streams.ini` (for Windows servers) or `streams.conf` (for Linux server), so you can also manually edit the default settings.

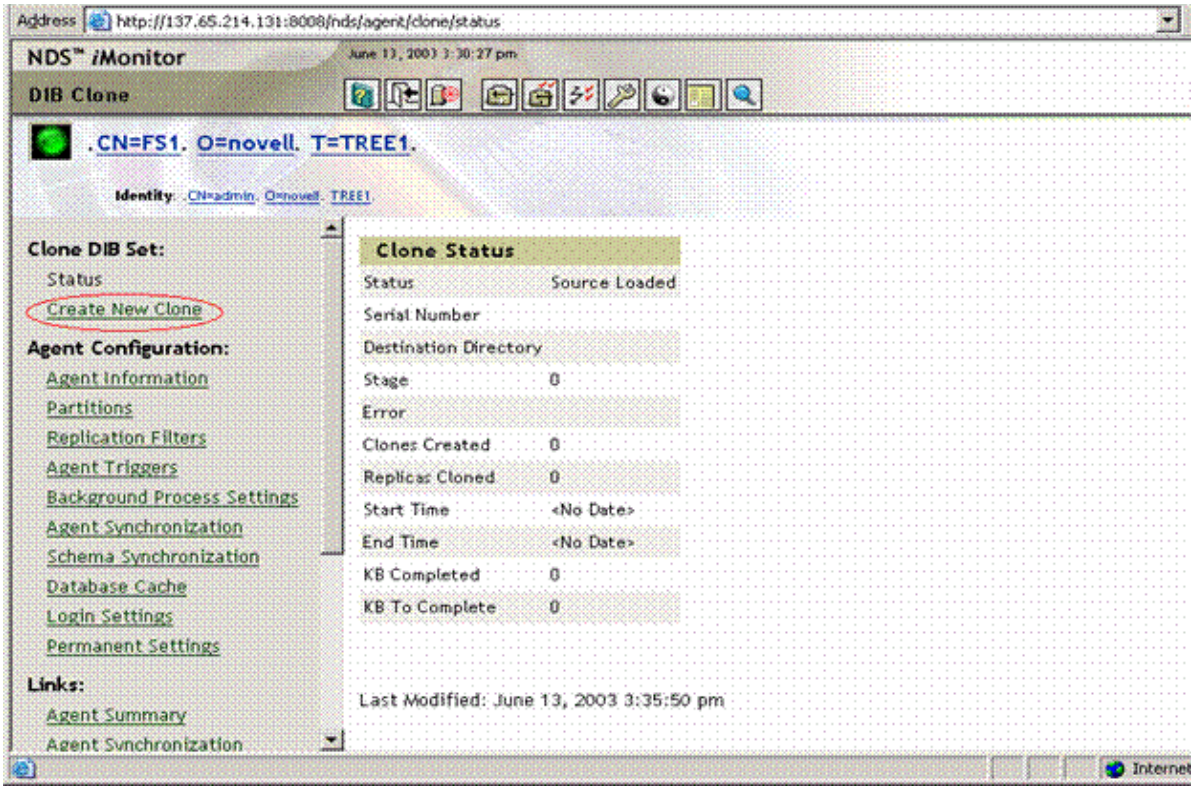
Clone DIB Set

This option creates a complete DIB fileset duplicate of an OpenText eDirectory database stored on a single server (the source server). The DIB Clone must be taken from the source server that holds all the master replicas in the tree. The clone can then be placed on another server (the target server). When the target server initiates OpenText eDirectory, it loads the DIB fileset, contacts the master replica of the server object, resolves its name, then synchronizes any changes to the DIB fileset made after the clone was created.

The clone of an OpenText eDirectory DIB set should only be placed on a server running the same operating system as the server the clone was created on. For example, if you want to restore a cloned DIB fileset to a Linux server, create the clone on a Linux server and not on a Windows server.

Although the back end for this feature was shipped with eDirectory 8.7, it was not supported until OpenText eDirectory 8.7.1 running OpenText iMonitor 2.4 or later. This option does not apply to any version of OpenText eDirectory or NDS prior to 8.7.

Figure 8-4 *Clone DIB Set Page in OpenText iMonitor*



This section includes the following information:

- "Clone DIB Set Use Cases" on page 243
- "Creating a Clone" on page 243

Clone DIB Set Use Cases

Clone DIB Set provides the following use cases:

- Create a new server with partitions already in an "on" state.

  Advantages include the following:

  - All servers in the ring do not need to be up and running to add a new server to the replica ring.
  - A new server will automatically have all partitions with no synchronization necessary.
  - Quicker up time.

- Disaster recovery

Advantages	Disadvantages
◆ Only need one copy of the partition to succeed. ◆ Less down time on large servers with multiple partitions.	◆ Must have at least one good copy of the partitions in question. ◆ Won't handle any SSL or security backups. ◆ Does not handle the file system.

- Backup and restore

Advantages	Disadvantages
◆ Quicker up time, especially on large scale databases.	◆ Only adds core OpenText eDirectory. LDAP, SNMP, SSL, etc. are not installed or configured. ◆ Will not get the latest changes. Only a snapshot is taken. Roll forward logs are not executed.

  Because of the listed disadvantages, we do not recommend using Clone DIB Set for backup and restore purposes.

Creating a Clone

A clone DIB fileset can be created with the originating server either online or offline. The offline method requires OpenText eDirectory to be brought down. In the online mode, OpenText eDirectory is up and not locked.

- "Online Method" on page 244
- "Offline Method with EBA Disabled" on page 245
- "Offline Method with EBA Enabled" on page 246
- "Completing the OpenText eDirectory Configuration" on page 248

WARNING: Do not use the Dibclone utility on an Identity Management server to clone another server, because this generates unnecessary TAO files on the cloned server.

Online Method

1 Load the ndsclone module on the source server.

Platform	To Extend the Schema
Windows	In NDSCons.exe, select **dsclone.dll**, then click **Start**.
Linux	Add an ndsclone entry to the ndsmodules.conf file, then use the http://IP address:port/dhost page to load the Directory Clone Agent. **NOTE:** The ndsclone module can also be loaded using the ndstrace -c "load ndsclone" command.

2 Disable the inbound sync from OpenText iMonitor agent configuration page before starting the clone DIB process on the source server.

3 Create the clone DIB fileset.

3a Run Clone DIB Configuration in OpenText iMonitor.

Click **Agent Configuration** > **Clone DIB Set** > **Create New Clone**.

3b Specify the fully qualified name of the target server and the file path where the cloned DIB files will be placed, then check the **Create Clone Object** and the **Clone DIB Online** boxes.

The NCP Server name (Clone Object) of the target server must match the target server name.

3c Click **Submit**.

The NDS Clone object is created and the DIB fileset is copied to the specified destination.

4 Install and configure OpenText eDirectory on the target server and bring down the server.

5 Copy the DIB directory containing the cloned DIB fileset to the target server.

Additionally, on Linux system, copy the /etc/opt/novell/eDirectory/conf/nds.conf file from the source server to the target server and update the following references to the target server:

- Change the IP Address for the following parameters
  - n4u.server.interfaces
  - http.server.interfaces
  - https.server.interfaces
- Provide the NCP Server Name which is created in step 3b in the n4u.nds.server-name parameter
- Provide the Preferred Server Name in n4u.nds.preferred-server parameter. Usually the host name of the target server is considered as the preferred server name.

6 Remove the nicisdi.key from /var/opt/novell/nici/0 and /var/opt/novell/nici/0/backup on the target server.

7 Now start the target server and run the ndsconfig upgrade command.

NOTE: On Windows, You need to run the EConfig.ps1 command to upgrade the OpenText eDirectory server using the silent installer. While upgrading, you need to mention the tree name, server name and admin credential of the cloned DIB in the upgrade.ni response file. You must also mention the existing server IP containing the IP of other servers in the tree. For more information, see Unattended Upgrade of OpenText eDirectory on Windows in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

8 Ensure that master replica of the target Server object is running OpenText eDirectory and is available. When OpenText eDirectory initializes on the target server, it communicates with the master replica where the final naming of the target server is resolved.

9 Make sure that the replica attribute value of the target server is synched with all the servers. Once the attribute changes are available on all servers, reenable the inbound sync on the source server. The inbound sync can be enabled either through the OpenText iMonitor agent configuration page or through DSTrace.

10 To complete the OpenText eDirectory configuration, see "Completing the OpenText eDirectory Configuration" on page 248.

Offline Method with EBA Disabled

1 Create the clone DIB fileset.

1a Run Clone DIB Configuration in OpenText iMonitor.

Click Agent Configuration > **Clone DIB Set** > **Create New Clone**.

1b Specify the fully qualified name of the target server, check the **Create Clone Object** box, then uncheck the **Clone DIB Online** box.

The NCP Server name of the target server must match the target server name.

1c Click **Submit**.

The NDS clone object is created, the DIB is locked in the source server, and an error reports that OpenText eDirectory is locked.

2 Install and configure OpenText eDirectory on the target server and bring down the server.

3 Manually copy the *.nds, nds*, and nds.rfl/*.* files from the source server's DIB directory to a destination or media on the target server convenient for moving the set to the target server's DIB directory. Additionally, on Linux system, transfer the /etc/opt/novell/eDirectory/conf/nds.conf file to the target server and update the following references to the target server:

- Change the IP Address for the following parameters
  - n4u.server.interfaces
  - http.server.interfaces
  - https.server.interfaces
- Provide the NCP Server Name which is created in step 1b in the n4u.nds.server-name parameter
- Provide the Preferred Server Name in n4u.nds.preferred-server parameter. Usually the host name of the target server is considered as the preferred server name.

4 Remove the nicisdi.key from /var/opt/novell/nici/0 and /var/opt/novell/nici/0/backup on the target server.

5 Export NDSD_DISABLE_INBOUND=Y environment variable, then start `ndsd` to disable the inbound sync on the source server.

6 Restart OpenText eDirectory on the source server.

If OpenText eDirectory is restarted on the source server before the files are copied, this clone is invalid. The new NCP Server object must then be deleted and the clone must be recreated.

7 Now start the target server and run the `ndsconfig upgrade` command.

NOTE: On Windows, You need to run the eDirectory Setup file. You also need to select and login to the eDirectory tree while the Setup file is being run to upgrade your OpenText eDirectory server.

---

8 Make sure that the replica attribute value of the target server is synched with all the servers. Once the attribute changes are available on all servers, reenable the inbound sync on the source server. The inbound sync can be enabled either through the OpenText iMonitor agent configuration page or through DSTrace.

9 Install OpenText eDirectory and start the server on the target server, with the DIB directory containing the cloned DIB fileset.

Ensure that master replica of the new target server object is running OpenText eDirectory and is available. When OpenText eDirectory initializes on the target server, it communicates with the master replica where the final naming of the target server is resolved.

10 To complete the OpenText eDirectory configuration, see "Completing the OpenText eDirectory Configuration" on page 248.

Offline Method with EBA Enabled

Prerequisites

- You must clone the DIB set from the master replica. The R/W replica is not supported while DIB cloning with EBA enabled. To determine the status of the master replica, run the following command in `/etc/opt/novell/eDirectory/conf` folder:

  ```
  ndsstat -r
  ```

- Run the `ndscheck` command on the source server to capture the status of EBA.

Performing DIB Cloning

1 Create the clone DIB fileset.

1a Run Clone DIB Configuration in OpenText iMonitor.

Click Agent Configuration > **Clone DIB Set** > **Create New Clone**.

1b Specify the fully qualified name of the target server, check the **Create Clone Object** box, then uncheck the **Clone DIB Online** box.

The NCP Server name of the target server must match the target server name.

1c Click **Submit**.

The NDS clone object is created, the DIB is locked in the source server, and an error reports that OpenText eDirectory is locked.

2 Install and configure OpenText eDirectory without enabling EBA on the target server and bring down the server.

3 Manually copy the `*.nds`, `nds*`, and `nds.rfl/*.*` files from the source server's DIB directory to a destination or media on the target server convenient for moving the set to the target server's DIB directory. Additionally, on Linux system, transfer the `/etc/opt/novell/eDirectory/conf/nds.conf` file to the target server and update the following references to the target server:

- Change the IP Address for the following parameters
  - n4u.server.interfaces
  - http.server.interfaces
  - https.server.interfaces
- Provide the NCP Server Name which is created in step 1b in the `n4u.nds.server-name` parameter
- Provide the Preferred Server Name in `n4u.nds.preferred-server` parameter. Usually the host name of the target server is considered as the preferred server name.
- Remove the parameter entry `n4u.server.eba_enabled` from the file.

4 Remove the `nicisdi.key` from `/var/opt/novell/nici/0` and `/var/opt/novell/nici/0/backup` on the target server.

5 Export NDSD_DISABLE_INBOUND=Y environment variable by adding the entry in the env file located at `/etc/opt/novell/eDirectory/conf` to disable inbound sync on the source server.

6 Restart OpenText eDirectory on the source server.

6a Run the `ndscheck` command to capture the EBA status.

6b Compare the status of EBA captured in prerequisites and step 6a. If there is any difference in status of EBA between both these occasions, restart eDirectory on the source server again.

6c Once OpenText eDirectory has been restarted on source server, run the `ndscheck` command again to ensure that the status of EBA is same as prerequisites.

NOTE: : If OpenText eDirectory is restarted on the source server before copying the files, the clone will be invalid. The new NCP Server object must then be deleted and the clone must be recreated.

---

7 Now start OpenText eDirectory on the target server and run the `ndsconfig upgrade` command.

7a When prompted to configure EBA, select `n`.

7b If `-601` error is encountered during the `ndsconfig upgrade`, then enable the sync on the source server by removing `NDSD_DISABLE_INBOUND=Y` from the source server and restart OpenText eDirectory on source server.

7c Once OpenText eDirectory is restarted on the source server, run the `ndsconfig upgrade` command on target server. Once You will be prompted to configure EBA, select `n`.

NOTE: On Windows, You need to run the OpenText eDirectory Setup file. You also need to select and login to the OpenText eDirectory tree while the Setup file is being run to upgrade your OpenText eDirectory server.

---

8 Run the `ndsconfig upgrade` command again and select the option y to enable EBA on target server.

Ensure that master replica of the new target server object is running OpenText eDirectory and is available. When OpenText eDirectory initializes on the target server, it communicates with the master replica where the final naming of the target server is resolved.

9 To complete the OpenText eDirectory configuration, see "Completing the OpenText eDirectory Configuration" on page 248.

Completing the OpenText eDirectory Configuration

- "SIDKEY" on page 237
- 'Configuring SAS, LDAP, and SNMP Services" on page 237

SDIKEY

1 Bring down OpenText eDirectory on the target server.

2 Move or rename the `/var/opt/novell/nici/0/nicisdi.key` and the `/var/opt/novell/nici/0/backup/nicisdi.key` file on file system of the target server.

Platform	Directory
Windows	`C:\Windows\SysWOW64\novell\nici\nicisdi.key`
Linux	`/var/opt/novell/nici/0/nicisdi.key` `/var/opt/novell/nici/0/backup/nicisdi.key`

3 Start OpenText eDirectory on the target server.

Configuring SAS, LDAP, HTTP, and SNMP Services

Linux: You can configure SAS, LDAP, SNMP, and HTTP services in one operation by entering the following command at the command line:

`ndsconfig upgrade [-a admin FDN]`

Windows: Run the OpenText eDirectory installer and complete the configuration of SAS, LDAP, SNMP, and HTTP services.

After completing the configuration, HTTP listens on ports 80 and 443 by default. OpenText eDirectory stores the HTTP port configuration on the HTTP server object. If required, you can change the port configuration as an administrator user.

For configuring the services individually, refer the following tables:

SAS

Platform	Command or Tool
Windows	Create SAS Service object and Certificates by using OpenText Identity Console.

LDAP

Platform	Command or Tool
Windows	Create LDAP Server and Group Objects by using OpenText Identity Console.

SNMP

Platform	Command or Tool
Windows	rundll32 snmpinst, snmpinst **-c createobj -a userFDN -p password -h hostname_or_IP_address**

Ensuring Secure OpenText iMonitor Operations

Securing access to your OpenText iMonitor environment involves the following protective steps:

1. Use a firewall and provide VPN access (this also applies to OpenText Identity Console and any other Web-based service that should have restricted access).
2. Whether a firewall is in place or not, limit the type of access allowed through OpenText iMonitor to further protect against Denial of Service (DoS) attacks.

   Although substantial efforts have been made to ensure that OpenText iMonitor validates the data it receives via URL requests, it is nearly impossible to guarantee that every conceivable invalid input is rejected. To reduce the risk of DoS attacks via invalid URLs, there are three levels of access that can be controlled through OpenText iMonitor's configuration file using the LockMask: option.

Access Level	Description
0	Require no authentication before OpenText iMonitor processes URLs. In this case, the OpenText eDirectory rights of the [Public] identity are applied to any request, and information displayed by iMonitor is restricted to the rights of the [Public] user. However, because no authentication is required to send URLs to OpenText iMonitor, it might be vulnerable to DoS attacks that are based on sending garbage in the URL.
1 (Default)	Before OpenText iMonitor processes URLs, require successful authentication as someOpenText eDirectory identity. In this case, the eDirectory rights of that identity are applied to any request and are, therefore, restricted by those rights. The same DoS vulnerability as level 0 exists, except the attack must be launched by someone who has actually authenticated to the server. Until a successful authentication occurs, the response to any OpenText iMonitor URL request is a login dialog box, so OpenText iMonitor should be impervious to attacks by unauthenticated users when it is configured in this state.

Access Level	Description
2	Before OpenText iMonitor processes URLs, require successful authentication as an OpenText eDirectory identity that has supervisor equivalency on the server that OpenText iMonitor is authenticating to. The same DoS vulnerability as level 1 exists, except the attack must now be launched by someone who has actually authenticated as a supervisor of the server. Until a successful authentication occurs, the response to any OpenText iMonitor URL request is a login dialog box, so OpenText iMonitor should be impervious to attacks by unauthenticated users and non-supervisor authenticated users when it is configured in this state.

Level 1 is the default because many administrators do not have supervisory access to every server in the tree but might need to use the OpenText iMonitor service on a server that their servers interact with.

NOTE: There are several features of OpenText iMonitor, such as Repair and Trace, that require supervisor equivalency to access regardless of the LockMask setting.

Configuring HTTP Server Object

An OpenText eDirectory installation creates an HTTP server object. The default configuration for HTTP Services is located in the directory on this object. However, you can modify the default configuration by using OpenText Identity Console. The HTTP server object represents server-specific configuration data.

The following are the attributes on the HTTP server object:

- **httpDefaultTLSPort:** Indicates the secure port at which HTTP the server listens.
- **httpDefaultClearPort:** Indicates the clear text port at which HTTP the server listens.
- **httpAuthRequiresTLS:** Indicates whether the request coming through the clear text port need to be redirected to a secure port.
- **httpTraceLevel:** Indicates the debug level of HTTP server in DSTrace. The default trace level is 2.
- **httpKeyMaterialObject:** Holds the DN of the certificate object which the HTTP server needs to use when handling the secure connection. To configure OpenText iMonitor interfaces in Suite B mode, enable the desired Suite B mode by setting the value of httpBindRestrictions to the Suite B mode and then associate an appropriate ECDSA server certificate to httpKeyMaterialObject.By default, httpkeyMaterialObject is set to use the RSA certificate.
- **httpSessionTimeout:** Indicates the timeout of the HTTP sessions. The default value is 900 seconds.
- **httpKeepAliveRequestTimeout:** Indicates the keep alive timeout of each HTTP request. The default value is 15 seconds.
- **httpRequestTimeout:** Indicates the timeout of each HTTP request. The default value is 300 seconds.
- **httpIOBufferSize:** Indicates the input and output buffer size of the HTTP server. The default value is 8192 bytes.
- **httpThreadsPerCPU:** Indicates the HTTP threads that has to be spawned per CPU. The default value is 2 threads.

- **httpHostServerDN:** Holds the DN of the NCP server object to which it is associated with.
- **httpBindRestrictions:** Allows you to set the cipher encryption level.
  - **RSA:** You can use the following values to restrict the cipher usage:
    - 0 - accept HIGH, MEDIUM, LOW and EXPORT ciphers
    - 1 - accept HIGH, MEDIUM, and LOW ciphers only
    - 2 - accept HIGH and MEDIUM ciphers only
    - 3 - accept HIGH ciphers only

    The default value is 3.
  - **ECDSA 256:** You can use the following value to restrict the cipher usage:
    - 4 - allows a 128-bit cipher or a 256-bit cipher
  - **ECDSA 384:** You can use the following values to restrict the cipher usage:
    - 5 - allows a 128-bit cipher or a 256-bit cipher
    - 6 - allows a 256-bit cipher

  For ECDSA certificates, OpenText eDirectory allows only Suite B ciphers.

  To configure LDAP and httpstk interfaces in Suite B mode, log in to OpenText Identity Console with administrator rights and enable one of the Suite B modes and then associate an appropriate ECDSA server certificate to these interfaces. You need to do this for every OpenText eDirectory server using the server's LDAP and httpstk configuration objects such as ldapServer and httpServer. Before turning on Suite B mode, ensure that all LDAP clients, LDAP browsers, and web browsers in the eDirectory environment support TLS 1.2 and EC certificates.

Setting HTTP Stack Parameters Using ndsconfig

The following are the HTTP stack parameters using ndsconfig:

- **http.server.interfaces:** Holds the clear text interface at which the HTTP server listens. This is set during a new instance configuration by ndsconfig.
- **http.server.request-io-buffer-size:** Indicates the input and output buffer size of the HTTP server. The detault value is 8192 bytes.
- **http.server.request_timeout-seconds:** Indicates the timeout of each HTTP request. The default value is 300 seconds.
- **http.server.keep-timeout-seconds:** Indicates the keep alive timeout of each HTTP request. The default value is 15 seconds.
- **http.server.threads-per-processor:** Indicates the HTTP threads that has to be spawned per CPU. The default value is 2 threads.
- **http.server.session-exp-seconds:** Indicates the time out of the HTTP sessions. The default value is 900 seconds.
- **http.server.trace-level:** Indicates the debugging level of HTTP stack in DSTrace. The default level is 2.
- **http.server.clear-port:** Indicates the clear text port at which HTTP server listens.
- **http.server.tls-port:** Indicates the secure port at which the HTTP server listens.
- **http.server.auth-req-tls:** Indicates whether the requests coming through clear text port need to be redirected to secure port.

- **https.server.interfaces:** Holds the secure interface at which the HTTP server listens. This is set during new instance configuration by ndsconfig.
- **https.server.cached-cert-dn:** Holds the DN of the certificate object, which the HTTP server needs to use while handling the secure connection.

Using cn=monitor for Monitoring

OpenText eDirectory provides an LDAP search method for monitoring the current state of an OpenText eDirectory server. OpenText eDirectory records useful performance metrics and server state information for OpenText eDirectory subsystems and background processes such as Threadpool, Connection Table, DClient, DS Agent, Background Processes, and LDAP Server as an entry with the base DN of `cn=monitor`. You can obtain the statistics from the server by issuing a search request with a search base of `cn=monitor` and use this information for monitoring your OpenText eDirectory environment.

IMPORTANT: `cn=monitor` is a virtual object and does not actually reside in the OpenText eDirectory tree. You can use this method for monitoring OpenText eDirectory through LDAP interfaces.

OpenText eDirectory subsystems are registered as data producers within the monitoring framework. Table 8-1 lists the registered data producers in OpenText eDirectory. The framework gathers real time data from all the registered data producers and shares it with the requestor of the data, who are the consumers of this data. The monitoring framework dynamically generates and returns objects in response to search requests in the `cn=monitor` subtree. Each object contains information about a particular aspect of the server. Some objects serve as containers for other objects and are used to construct a hierarchy of objects, where `cn=monitor` is the most superior object. You can use LDAP clients to access information provided by the monitoring framework, subject to access and other controls, such as LDAP server specific information or connection-specific information. OpenText eDirectory restricts this search request only to users with write rights to the NDSRightsToMonitor attribute on the NCP server object.

You can access data from all the registered data producers with ldapsearch or with any general-purpose LDAP browser.

To view the monitoring data from all the registered data producers, use the ldapsearch command:

```
ldapsearch -h <SrvIP> -p <port> -D <user dn> -w <password> -s sub -b
cn=monitor
```

NOTE: OpenText eDirectory does not support data filtering on `cn=monitor` search. For some background processes that are scheduled to run recursively, OpenText eDirectory displays these processes multiple times as scheduled in the `cn=monitor` search response. For example, SkulkerWorkerProc.

Viewing the Monitoring Statistics

`ldapsearch` returns data from all the registered data producers in LDAP format using `cn=monitor` as a base. The LDAP server also acts as a data consumer in the LDAP object format.

Table 8-1 lists the data producers and the corresponding parameters containing the monitoring statistics. There could be additional data producers when other products are configured with OpenText eDirectory.

Table 8-1 *Data Producers and Monitoring Statistics Parameters*

Data Producers	Monitoring Statistics Parameters
Agent	◆ Background Process ◆ Partition ◆ System State
DHOST	The following DHOST processes and connection information is monitored: ◆ Inbound connections ◆ Thread pool information ◆ ThreadsSpawned ◆ ThreadsDied ◆ ThreadsIdle ◆ ThreadsWorkers ◆ ThreadPeakWorkers ◆ ThreadPoolReadyQueueItems ◆ ThreadPoolReadyQueueMaxWaitTime ◆ ThreadMinWaitTime ◆ ThreadMaxWaitTime
DClient	◆ Outbound context ◆ Outbound connection
LDAP	◆ Binding ◆ Errors ◆ Incoming operations ◆ Outgoing operations ◆ Replications ◆ Traffic volume

Data Producers	Monitoring Statistics Parameters
Record Manager	◆ CacheFault Looks ◆ Cache Faults ◆ Current Size,Hits ◆ Hit Looks ◆ Item Cached ◆ Maximum size ◆ OldVersionCachedCount ◆ OldVersionCachedSize ◆ DIb Size ◆ Checkpoint thread

When you issue a search request with a search base of `cn=monitor`, the monitoring framework dynamically generates and returns objects in response to the search request in the cn=monitor subtree as listed in Table 8-2.

Table 8-2 *Objects Monitored by cn=monitor Search*

Object Name	Description
cn=Monitor	Root level object for monitoring data.
cn=Agent,cn=Monitor	Provides information about Directory Service agent.
cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about background process. (is this any specific process or in general all b/g processes)
cn=ARC resolve timer thread,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about advanced referral costing background process.
cn=BacklinkProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about backlinker background process.
cn=CPU Usage monitor,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about CPU usage background process.
cn=CheckBacklinks,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about checking the backlinker background process.
cn=CheckExtRefProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about checking external reference background process.
cn=ExtRefRefreshProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about refresh external references background process.
cn=Janitor,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about Janitor background process.

Object Name	Description
cn=RunLimberUp,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about schedule limber background process.
cn=Limber,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about Limber (connectivity check) background process.
cn=HiConvergenceHeartBeat,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about schedule skulker background process.
cn=ObitProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about Obituary background process.
cn=PartitionPurgeProcess,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about partition purger background process.
cn=Predicate Statistics Update,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about predicate statistics update background process.
cn=RNRAdvertise,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about advertise service address background process.
cn=RefreshBinderyContext,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about refresh bindery background process.
cn=Repair Inactive Replicas,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about repair inactive replicas background process.
cn=SchemaProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about schema sync background process.
cn=SkulkerProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about synchronization background process.
cn=SkulkerWorkerProc,cn=BackGroundProcInterval,cn=Agent,cn=Monitor	Provides information about synchronization background process.
cn=Partition,cn=Agent,cn=Monitor	Provides information about all user partitions on the server. Multiple values of the same attribute denote multiple partitions.
cn=Status,cn=Agent,cn=Monitor	Provides information about server status.
cn=DHOST,cn=Monitor	Provides information about DHOST subsystems.
cn=InBoundConnection,cn=DHOST,cn=Monitor	Provides information about inbound connection table information.
cn=ThreadPool,cn=DHOST,cn=Monitor	Provides information about DHOST Threadpool statistics.
cn=Dclient,cn=Monitor	Provides information about server-side DClient.
cn=OutBoundConnection,cn=Dclient,cn=Monitor	Provides information about outbound connection table information.
cn=OutBoundContext,cn=Dclient,cn=Monitor	Provides information about outbound context table information.

Object Name	Description
cn=LDAP,cn=Monitor	Provides information on LDAP server.
cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about LDAP server statistics.
cn=Bindings,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about binding statistics on LDAP server.
cn=Errors,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about the errors that occurred during the LDAP request.
cn=IncomingOperations,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about incoming operation statistics on LDAP server.
cn=OutgoingOperations,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about outgoing operations statistics on LDAP server.
cn=Replications,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about the LDAP server replication statistics.
cn=TrafficVolume,cn=LDAPStatistics,cn=LDAP,cn=Monitor	Provides information about LDAP server traffic volume statistics.
cn=RecordManager,cn=Monitor	Provides information about FLAIM database.
cn=Size,cn=RecordManager,cn=Monitor	Provides information about size information of the FLAIM database.
cn=CheckPointThreadData,cn=RecordManager,cn=Monitor	Provides information about check point thread.
cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides information about FLAIM database cache statistics.
cn=CacheFaultLooks,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides cache fault looks information.
cn=CacheFaults,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides cache fault information.
cn=HitLooks,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides cache hit looks information.
cn=Hits,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides cache hits information.
cn=ItemsCached,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides number of items cached information.
cn=OldVersionCachedCount,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides old version cached item count information.
cn=MaximumSize,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides maximum cache size information.
cn=CurrentSize,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides current cache size information.
cn=OldVersionCachedSize,cn=CacheStatistics,cn=RecordManager,cn=Monitor	Provides old version cached size information.

Each object contains information about a particular aspect of the server, such as a connection or a thread. Table 8-3 lists the attributes that hold the monitoring statistics.

Table 8-3 *Attributes Monitoring Statistics*

Attribute	Description
BackgroundProcScheduled	Next scheduled time for the background process. Multiple values denote that the background process is scheduled multiple times.
BackgroundProcStartTime	Next start time for the background process. Multiple values denote that the background process is running multiple times.
PerishableData	The amount of data not synced out to any other server (denoted in seconds).
OBIT_NEWRDN_PURGEABLE	Number of NEWRDN obituaries in purgeable state.
OBIT_NEWRDN_OK_TO_PURGE	Number of NEWRDN obituaries in ok to purge state.
OBIT_NEWRDN_NOTIFIED	Number of NEWRDN obituaries in notified state.
OBIT_NEWRDN_ISSUED	Number of NEWRDN obituaries in issued state.
OBIT_MOVED_PURGEABLE	Number of moved obituaries in purgeable state.
OBIT_MOVED_OK_TO_PURGE	Number of moved obituaries in ok to purge state.
OBIT_MOVED_NOTIFIED	Number of moved obituaries in notified state.
OBIT_MOVED_ISSUED	Number of moved obituaries in issued state.
OBIT_DEAD_PURGEABLE	Number of dead obituaries in purgeable state.
OBIT_DEAD_OK_TO_PURGE	Number of dead obituaries in ok to purge state.
OBIT_DEAD_NOTIFIED	Number of dead obituaries in notified state.
OBIT_DEAD_ISSUED	Number of dead obituaries in issued state.
OBIT_COUNT_FROM_DATABASE_INDEX	Total obituary count.
MaxRingDelta	Maximum amount of data not synchronized between any two servers in the replica ring (denoted in seconds).
ChangeCacheCount	Current change cache count on the partition.
eDirectoryUpTime	Number of seconds since server has been running.
eDirectorySystemCurrTime	Current system time of the server.
eDirectoryAgentVersion	Current Directory Server agent version.
MaxInBoundConnection	Maximum inbound connection.
InBoundConnectionCount	Current inbound connection count.
ThreadsWorkers	Number of worker threads in Threadpool.
ThreadsSpawned	Number of threads spawned.

Attribute	Description
ThreadsIdle	Number of threads idle.
ThreadsDied	Number of threads died.
ThreadWaitingQueuePeakItems	Maximum number of threads in the waiting queue.
ThreadWaitingQueueItems	Current number of threads in the waiting queue.
ThreadPoolReadyQueueMaxWaitTime	Maximum wait time for thread in ReadyQueue.
ThreadPoolReadyQueueItems	Current number of threads in ReadyQueue.
ThreadPeakWorkers	Maximum number of pool workers.
ThreadMinWaitTime	Minimum thread wait time before getting scheduled.
ThreadMaxWaitTime	Maximum thread wait time before getting scheduled.
TotalOpenOutBoundConnection	Current open outbound connection count.
RefusedOutBoundConnection	Refused outbound connection count.
MaxOutBoundConnection	Maximum outbound connection count.
TotalOutBoundContextCount	Maximum outbound context count.
ActiveOutBoundContextCount	Current outbound context count.
unAuthBinds	Number of unauthenticated/anonymous bind requests received.
strongAuthBinds	Number of bind requests that were authenticated using SASL and X.500 strong authentication procedures. This includes the binds that were authenticated using external authentication procedures.
simpleAuthBinds	Number of bind requests that were authenticated using simple authentication procedures where the password is sent over the wire in encrypted or clear text format.
bindSecurityErrors	Number of bind requests that have been rejected due to inappropriate authentication or invalid credentials.
Errors	Number of requests returned with an error. The following are not included under Errors attribute: ◆ Security related errors ◆ Referral related errors ◆ Partially returned requests
securityErrors	Number of requests that did not meet the security requirements, such as inappropriate authentication or invalid credentials.
wholeSubtreeSearchOps	Number of whole subtree search requests received.
searchOps	Number of search requests (baseObject searches, oneLevel searches, and whole subtree searches) received.
removeEntryOps	Number of removeEntry requests received.

Attribute	Description
readOps	Number of read requests received.
oneLevelSearchOps	Number of oneLevel search requests received.
modifyRDNOps	Number of modifyRDN requests received.
modifyEntryOps	Number of modifyEntry requests received.
listOps	Number of list requests received.
inOps	Number of requests received from client.
extendedOps	Number of extended operations.
compareOps	Number of compare requests received.
addEntryOps	Number of addEntry requests received.
abandonOps	Number of LDAP abandoned requests.
referralsReturned	Number of referrals returned in response to requests for operations.
chainings	Number of operations forwarded by this OpenText eDirectory server to other OpenText eDirectory servers.
repsUpdatesIn	Current inbound replication update requests count.
repsUpdateOut	Current outbound replication update requests count.
outBytes	Outgoing traffic, in bytes, on the interface. This includes responses to client and OpenText eDirectory servers as well as requests to other OpenText eDirectory servers.
inBytes	Incoming traffic, in bytes, on the interface. This includes requests from client as well as responses from other OpenText eDirectory servers.
Total	Total item count in FLAIM cache.
EntryCache	Total item count in entry cache.
BlockCache	Total item count in block cache.
TotalSize	Total item size in FLAIM cache.
EntryCacheSize	Total item size in entry cache.
BlockCacheSize	Total item size in block cache.
CheckPointThreadWritingDataBlocks	0 denotes that checkpoint is not writing dirty blocks. 1 denotes that the checkpoint is writing dirty blocks.
CheckPointThreadStartTime	Checkpoint thread start time. Look at this value only if checkpoint thread is running.
CheckPointThreadLogBlocksWritten	Number of log blocks written.
CheckPointThreadIsRunning	0 denotes that the checkpoint is not running. 1 denotes that checkpoint thread is running.
CheckPointThreadIsForced	Denotes whether checkpoint was forced.

Attribute	Description
CheckPointThreadForceStartTime	Checkpoint forced start time. Look at this value only if checkpoint is forced started.
CheckPointThreadDirtyCacheBlocks	Number of dirty cache blocks.
CheckPointThreadDataBlocksWritten	Number of dirty blocks written.
CheckPointThreadBlockSize	Current block size.
TotalDIBSize	Total FLAIM database size.
DIBStreamFileSize	Total stream files size.
DIBRollBackFileSize	Total roll back files size.
DIBRflmFileSize	Total roll forward log files size.
DIBFileSize	Total DIB files size.
CurrentTransactionID	The Transaction ID is used to keep track of changes that happen in the database. Each `write` operation happens within a transaction (associated with a `transactionID`). The maximum allowed transaction ID is 0xFFFFE000 [4294959104]. At this point, no new transactions will be allowed.

A sample LDAP search output is below.

```
# LDAPv3
# base <cn=monitor> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
# BackGroundProcInterval, Agent, Monitor
dn: cn=BackGroundProcInterval,cn=Agent,cn=Monitor
slowSyncInterval: 1800
fastSyncInterval: 5
ServerStateUpThreshold: 1800
JanitorInterval: 120
HeartBeatSkulkInterval: 3600
FlatCleaningInterval: 43200
DRLInterval: 60
BacklinkInterval: 46800
objectclass: Top
objectclass: extensibleObject
# .GOOD-ONE., Partition, Agent, Monitor
dn: cn=.GOOD-ONE.,cn=Partition,cn=Agent,cn=Monitor
ChangeCacheCount: 0
objectclass: Top
objectclass: extensibleObject
# InBoundConnection, DHOST, Monitor
dn: cn=InBoundConnection,cn=DHOST,cn=Monitor
MaxInBoundConnection: 256
InBoundConnectionCount: 20
objectclass: Top
objectclass: extensibleObject
# ThreadPool, DHOST, Monitor
```

```
dn: cn=ThreadPool,cn=DHOST,cn=Monitor
ThreadsWorkers: 37
Monitoring
ThreadsSpawned: 3572
ThreadsIdle: 7
ThreadsDied: 3535
ThreadWaitingQueuePeakItems: 24
ThreadWaitingQueueItems: 20
ThreadPoolReadyQueueMaxWaitTime: 574529
ThreadPoolReadyQueueItems: 0
ThreadPeakWorkers: 90
ThreadMinWaitTime: 2
ThreadMaxWaitTime: 16394616
objectclass: Top
objectclass: extensibleObject
# OutBoundConnection, Dclient, Monitor
dn: cn=OutBoundConnection,cn=Dclient,cn=Monitor
TotalOpenOutBoundConnection: 17
RefusedOutBoundConnection: 0
MaxOutBoundConnection: 4294967295
objectclass: Top
objectclass: extensibleObject
# OutBoundContext, Dclient, Monitor
dn: cn=OutBoundContext,cn=Dclient,cn=Monitor
TotalOutBoundContextCount: 256
objectclass: Top
objectclass: extensibleObject
# Bindings, LDAPStatistics, LDAP, Monitor
dn: cn=Bindings,cn=LDAPStatistics,cn=LDAP,cn=Monitor
unAuthBinds: 6908
strongAuthBinds: 0
simpleAuthBinds: 4433475
bindSecurityErrors: 0
objectclass: Top
objectclass: extensibleObject
# Errors, LDAPStatistics, LDAP, Monitor
dn: cn=Errors,cn=LDAPStatistics,cn=LDAP,cn=Monitor
securityErrors: 0
errors: 1
objectclass: Top
objectclass: extensibleObject
# IncomingOperations, LDAPStatistics, LDAP, Monitor
dn: cn=IncomingOperations,cn=LDAPStatistics,cn=LDAP,cn=Monitor
wholeSubtreeSearchOps: 4426462
searchOps: 4426462
removeEntryOps: 0
readOps: 0
oneLevelSearchOps: 0
modifyRDNOps: 0
modifyEntryOps: 4
listOps: 0
inOps: 8901739
extendedOps: 0
compareOps: 0
addEntryOps: 5
```

```
abandonOps: 0
objectclass: Top
objectclass: extensibleObject
# OutgoingOperations, LDAPStatistics, LDAP, Monitor
dn: cn=OutgoingOperations,cn=LDAPStatistics,cn=LDAP,cn=Monitor
chainings: 0
referralsReturned: 0
objectclass: Top
objectclass: extensibleObject
# Replications, LDAPStatistics, LDAP, Monitor
dn: cn=Replications,cn=LDAPStatistics,cn=LDAP,cn=Monitor
repUpdatesIn: 0
repUpdatesOut: 0
objectclass: Top
objectclass: extensibleObject
# TrafficVolume, LDAPStatistics, LDAP, Monitor
dn: cn=TrafficVolume,cn=LDAPStatistics,cn=LDAP,cn=Monitor
outBytes: 326809576
inBytes: 380249498
objectclass: Top
objectclass: extensibleObject
Monitoring
# CacheFaultLooks, RecordManager, Monitor
dn: cn=CacheFaultLooks,cn=RecordManager,cn=Monitor
TotalSize: 2699
EntryCacheSize: 2539
BlockCacheSize: 160
objectclass: Top
objectclass: extensibleObject
# CacheFaults, RecordManager, Monitor
dn: cn=CacheFaults,cn=RecordManager,cn=Monitor
TotalSize: 1948
EntryCacheSize: 1788
BlockCacheSize: 160
objectclass: Top
objectclass: extensibleObject
# CurrentSize, RecordManager, Monitor
dn: cn=CurrentSize,cn=RecordManager,cn=Monitor
TotalSize: 4849664
EntryCacheSize: 3866624
BlockCacheSize: 983040
objectclass: Top
objectclass: extensibleObject
# HitLooks, RecordManager, Monitor
dn: cn=HitLooks,cn=RecordManager,cn=Monitor
TotalSize: 656418775
EntryCacheSize: 489811630
BlockCacheSize: 166607145
objectclass: Top
objectclass: extensibleObject
# Hits, RecordManager, Monitor
dn: cn=Hits,cn=RecordManager,cn=Monitor
TotalSize: 449815580
EntryCacheSize: 283226835
BlockCacheSize: 166588745
```

```
objectclass: Top
objectclass: extensibleObject
# ItemsCached, RecordManager, Monitor
dn: cn=ItemsCached,cn=RecordManager,cn=Monitor
TotalSize: 1865
EntryCacheSize: 1691
BlockCacheSize: 174
objectclass: Top
objectclass: extensibleObject
# MaximumSize, RecordManager, Monitor
dn: cn=MaximumSize,cn=RecordManager,cn=Monitor
TotalSize: 200015872
EntryCacheSize: 100007972
BlockCacheSize: 100007900
objectclass: Top
objectclass: extensibleObject
# OldVersionCachedCount, RecordManager, Monitor
dn: cn=OldVersionCachedCount,cn=RecordManager,cn=Monitor
TotalSize: 7
EntryCacheSize: 3
BlockCacheSize: 4
objectclass: Top
objectclass: extensibleObject
# OldVersionCachedSize, RecordManager, Monitor
dn: cn=OldVersionCachedSize,cn=RecordManager,cn=Monitor
Monitoring
TotalSize: 21376
EntryCacheSize: 4448
BlockCacheSize: 16928
objectclass: Top
objectclass: extensibleObject
dn: cn=Size,cn=RecordManager,cn=Monitor
CurrentTransactionID: 26552
TotalDIBSize: 1533393 Bytes
DIBStreamFileSize: 781 Bytes
DIBRflmFileSize: 512708 Bytes
DIBRollBackFileSize: 167936 Bytes
DIBFileSize: 851968 Bytes
objectclass: Top
objectclass: extensibleObject
# search result
search: 2
result: 0 Success
# numResponses: 20
# numEntries: 19
```

Using DSTrace

To use the DSTrace utility in a Linux environment, run the following command from the server prompt:

```
/opt/novell/eDirectory/bin/ndstrace
```

The full syntax for the ndstrace command is as follows:

```
ndstrace [-l|-u|-c "command1;....."|--version] [-h
<local_interface:port>] [--config-file <configuration_file_path>] [thrd
<thread ID>] [svty <severity_level>] [conn <connection_ID>]
```

The DSTrace utility has three main parts:

- “Basic Functions” on page 264
- “Debugging Messages” on page 265
- “Background Processes” on page 267

Basic Functions

The basic functions of DSTrace are to:

- View internal OpenText eDirectory activity and debugging messages in Linux.
- Initiate limited synchronization processes.

You can use the DSTrace utility in either UI mode or command line mode. By default, DSTrace runs in UI mode. To start DSTrace in UI mode, enter the following command at the server prompt:

```
/opt/novell/eDirectory/bin/ndstrace
```

To start DSTrace in command line mode, enter the following command at the prompt:

```
/opt/novell/eDirectory/bin/ndstrace -l
```

To initiate basic DSTrace functions, enter commands at the server prompt using the following syntax:

```
ndstrace command_option
```

The following table lists the command options that you can enter.

Option	Description
ON	Starts the OpenText eDirectory trace screen with basic trace messages.
OFF	Disables the trace screen.
ALL	Starts the OpenText eDirectory trace screen and displays all the trace messages.
AGENT	Starts the OpenText eDirectory trace screen with the trace messages that are equivalent to the ON, BACKLINK, DSAGENT, JANITOR, RESNAME, and VCLIENT flags.
DEBUG	Turns on a predefined set of trace messages typically used for debugging. The flags set are ON, BACKLINK, ERRORS, EMU, FRAGGER, INIT, INSPECTOR, JANITOR, LIMBER, MISC, PART, RECMAN, REPAIR, SCHEMA, SKULKER, STREAMS, and VCLIENT.
NODEBUG	Leaves the trace screen enabled, but turns off all debugging messages previously set. This option also leaves the messages set to the ON command option.

Debugging Messages

When the DSTrace screen is enabled, the information displayed is based on a default set of filters. If you want to view more or less than the default, you can manipulate the filters using the debugging message flags. The debugging messages help you determine the status of OpenText eDirectory and verify that everything is working well.

Each OpenText eDirectory process has a set of debugging messages. To view the debugging messages on a particular process, use a plus sign (+) and the process name or option. To disable the display of a process, use a minus sign (-) and the process name or option. The following are some examples:

Message	Description
`set ndstrace = +SYNC`	Enables the synchronization messages.
`set ndstrace = -SYNC`	Disables the synchronization messages.
`set ndstrace = +SCHEMA`	Enables the schema messages.

You can also combine the debugging message flags by using the Boolean operators & (which means AND) and | (which means OR). The syntax for controlling the debugging messages at the server console is as follows:

`set ndstrace = <trace_flag> [parameter]`

The following table describes the trace flags for the debugging messages. You can enter abbreviations for each of the trace flags.

Trace Flag	Description
ABUF	Messages and information related to inbound and outbound packet buffers that contain data being received in conjunction with, or in response to, an OpenText eDirectory request.
ALOC	Messages to show the details of memory allocation.
AREQ	Messages related to inbound requests from other servers or clients.
AUTH	Messages and error reports relating to authentication.
BASE	Debug error messages at the minimum debugging level.
BLNK	Backlink and inbound obituary messages and error reports.
CBUF	Messages related to outbound DS Client requests.
CHNG	Change cache messages.
COLL	Status and error reports concerning an object's update information when the update has been previously received.
CONN	Messages that show information about the servers your server is trying to connect to, and about errors and timeouts that might be causing your server not to connect.
DNS	Messages about the OpenText eDirectory-integrated DNS server processes.

Trace Flag	Description
DRLK	Distributed reference link messages.
DVRS	Messages to show DirXML® driver-specific areas that OpenText eDirectory might be working on.
DXML	Messages to show details of DirXML events.
FRAG	Messages from the NCP™ fragger which breaks OpenText eDirectory messages into NCP-sized messages.
IN	Messages related to inbound requests and processes.
INIT	Messages related to the initialization of OpenText eDirectory.
INSP	Messages related to the integrity of objects in the source server's local database. Using this flag increases the demands on the source server's disk storage system, memory, and processor. Do not leave this flag enabled unless objects are being corrupted.
JNTR	Messages related to the following background processes: janitor, replica synchronization, and flat cleaner.
LDAP	Messages related to the LDAP server.
LMBR	Messages related to the limber process.
LOCK	Messages related to the use and manipulation of the source server's local database locks.
LOST	Messages related to lost entries.
MISC	Messages from different sources in OpenText eDirectory.
MOVE	Messages from the move partition or move subtree operations.
NCPE	Messages to show the server receiving NCP-level requests.
NMON	Messages related to OpenText iMonitor.
OBIT	Messages from the obituary process.
PART	Messages related to partition operations from background processes and from request processing.
PURG	Messages about the purge process.
RECM	Messages related to the manipulation of the source server's database.
RSLV	Reports related to the processing of resolve name requests.
SADV	Messages related to the registration of tree names and partitions with Service Location Protocol (SLP).
SCMA	Messages related to the schema synchronization process.
SCMD	Messages showing the details of schema-related operations. They give details of both inbound and outbound synchronization.
SKLK	Messages related to the replica synchronization process.
SPKT	Messages related to OpenText eDirectory NCP server-level information.

Trace Flag	Description
STRM	Messages related to the processing of attributes with a Stream syntax.
SYDL	Messages showing more details during the replication process.
SYNC	Messages about inbound synchronization traffic (what is being received by the server).
TAGS	Displays the tag string that identifies the trace option that generated the event on each line displayed by the trace process.
THRD	Messages to show when any background processes (threads) begin and end.
TIME	Messages about the transitive vectors that are used during the synchronization process.
TVEC	Messages related to the following attributes: Synchronize Up To, Replica Up To, and Transitive Vector.
VCLN	Messages related to the establishment or deletion of connections with other servers.

As you use the debugging messages in DSTrace, you will find that some of the trace flags are more useful than others. One of the favorite DSTrace settings of OpenText Support is actually a shortcut:

```
set ndstrace = A81164B91
```

This setting enables a group of debugging messages.

Background Processes

In addition to the debugging messages, which help you check the status of OpenText eDirectory, there is a set of commands that force the OpenText eDirectory background processes to run. To force the background process to run, place an asterisk (*) before the command. For example:

```
set ndstrace = *H
```

You can also change the status, timing, and control for a few of the background processes. To change these values, place an exclamation point (!) before the command and enter a new parameter or value. For example:

```
set ndstrace = !H 15 (parameter_value_in_minutes)
```

The following is the syntax for each statement controlling the background processes of OpenText eDirectory:

```
set ndstrace = <trace_flag> [parameter]
```

The following table lists the trace flags for the background processes, any required parameters, and the process the trace flags are displayed.

Trace Flag	Parameters	Description
*A	None	Resets the address cache on the source server.
*AD	None	Disables the address cache on the source server.

Trace Flag	Parameters	Description
*AE	None	Enables the address cache on the source server.
*B	None	Schedules the backlink process to begin execution on the source server in one second.
!B	Time	Sets the interval (in minutes) for the backlink process. Default=1500 minutes (25 hours) Range=2 to 10080 minutes (168 hours)
*CT	None	Displays the source server's outbound connection table and the current statistical information for the table. These statistics do not give any information about the inbound connections from other servers or clients to the source server.
*CTD	None	Displays, in comma-delimited format, the source server's outbound connection table and the current statistical information for the table. These statistics do not give any information about the inbound connections from other servers or clients to the source server.
*D	Replica rootEntry ID	Removes the specified local entry ID from the source server's Send All Object list. The entry ID must specify a partition root object that is specific to the server's local database. This command is usually used only when a Send All Updates process is endlessly trying to show updates and failing because a server is inaccessible.
!D	Time	Sets the inbound and outbound synchronization interval to the specified number of minutes. Default=24 minutes. Range=2 to 10080 minutes (168 hours)
!DI	Time	Sets the inbound synchronization interval to the specified number of minutes. Default=24 minutes Range=2 to 10080 minutes (168 hours)
!DO	Time	Sets the outbound synchronization interval to the specified number of minutes. Default=24 minutes Range=2 to 10080 minutes (168 hours)
*E	None	Reinitializes the source server's entry cache.
!E	None	Schedules the inbound and outbound synchronization processes to begin execution.
!EI	None	Schedules the inbound synchronization process to begin execution.
!EO	None	Schedules the outbound synchronization process to begin execution.
*F	None	Schedules the flat cleaner process, which is part of the janitor process, to begin execution on the source server in five seconds.

Trace Flag	Parameters	Description
!F	Time	Sets the interval (in minutes) for the flat cleaner process. Default=240 minutes (4 hours) Range=2 to 10080 minutes (168 hours)
*FL	1-10	Sets the number of rolling log files used by DSTrace. If you set this parameter to any value greater than 1, once the source server's `ndstrace.log` file reaches the configured maximum file size, DSTrace renames the file `ndstrace1.log` and creates a new `ndstrace.log` file. When that file reaches its maximum file size, the previous `ndstrace1.log` file is renamed `ndstrace2.log`, and the more recent `ndstrace.log` file is renamed `ndstrace1.log`. This process continues until DSTrace reaches the maximum number of rolling log files set by this option. Once the specified limit is reached, the oldest log files will be deleted and only the specified maximum number of rolling files will be maintained. You can configure a maximum of 10 rolling log files. By default, DSTrace must use at least 1 rolling log file. If you set this parameter to 0, DSTrace uses 1 as the parameter value.
*G	Replica rootEntry ID	Rebuilds the change cache of the specified root partition ID.
*H	None	Schedules the replica synchronization process to begin execution immediately on the source server.
!H	Time	Sets the interval (in minutes) for the heartbeat synchronization process. Default=30 minutes Range=2 to 1440 minutes (24 hours)
*HR	None	Clears the in-memory last-sent vector.
*I	Replica rootEntry ID	Adds the specified local entry ID to the source server's Send All Object list. The entry ID must specify a partition root object that is specific to the server's local database. The replica synchronization process checks the Send All Object list. If the entry ID of a partition's root object is in the list, OpenText eDirectory synchronizes all objects and attributes in the partition, regardless of the value of the Synchronized Up To attribute.
!I	Time	Sets the interval (in minutes) for the heartbeat synchronization process. Default=30 minutes Range=2 to 1440 minutes (24 hours)
*J	None	Schedules the purge process, which is part of the replica synchronization process, to begin running on the source server.

Trace Flag	Parameters	Description
!J	Time	Sets the interval (in minutes) for the janitor process. Default=2 minutes Range=1 to 10080 minutes (168 hours)
*L	None	Schedules the limber process to begin running on the source server in five seconds.
*M	Bytes	Changes the maximum file size used by the source server's `ndstrace.log` file. The command can be used regardless of the state of the debug file. The bytes specified must be a decimal value between 10000 bytes and 100 MB. If the value specified is higher or lower than the specified range, no change occurs.
!M	None	Reports the maximum memory used by OpenText eDirectory.
!N	0|1	Sets the name form. 0=hex only 1=full dot form
*P	None	Displays the tunable parameters and their default settings.
*R	None	Resets the size of the `ndstrace.log` file to zero bytes. This command is the same as the SET parameter NDS Trace File Length Set to Zero.
*S	None	Schedules the Skulker process, which checks whether any of the replicas on the server need to be synchronized.
!SI	Time	Sets the interval (in minutes) for the inbound schema synchronization process. Default=24 minutes Range=2 to 10080 minutes (168 hours)
!SO	Time	Sets the interval (in minutes) for the outbound schema synchronization process. Default=24 minutes Range=2 to 10080 minutes (168 hours)
!SIO	Time	Disables the inbound schema synchronization process for the specified number of minutes. Default=24 minutes Range=2 to 10080 minutes (168 hours)
!SO0	Time	Disables the inbound schema synchronization process for the specified number of minutes. Default=24 minutes Range=2 to 10080 minutes (168 hours)
*SS	None	Forces immediate schema synchronization.
*SSA	None	Schedules the schema synchronization process to begin immediately and forces schema synchronization with all target servers, even if they have been synchronized in the last 24 hours.

Trace Flag	Parameters	Description
*SSD	None	Resets the source server's Target Schema Sync list. This list identifies which servers the source server should synchronize with during the schema synchronization process. A server that does not hold any replicas sends a request to be included in the target list of a server that contains a replica with its Server object.
*SSL	None	Prints the schema synchronization list of target servers.
*ST	None	Displays the status information for the background processes on the source server.
*STX	None	Displays the status information for the backlink process (external references) on the source server.
*STS	None	Displays the status information for the schema synchronization process on the source server.
*STO	None	Displays the status information for the backlink process (obituaries) on the source server.
*STL	None	Displays the status information for the limber process on the source server.
!T	Time	Sets the interval (in minutes) for checking the server's UP state. Default=30 minutes Range=1 to 720 minutes (12 hours)
*U	Optional ID of server	If the command does not include an entry ID, this changed the status of any server that has been previously labeled **down** to **up**. If the command includes a local entry ID, it changes the status of the specified server from **down** to **up**. Entry IDs are specific to the source server's database and must refer to an object that represents a server.
!V	A list	Lists the restricted OpenText eDirectory versions. If no versions are listed, there are no restrictions. Each version is separated by a comma.
*Z	None	Displays the currently scheduled tasks.

DSTrace Messages

You can filter the trace messages based on the thread ID, connection ID, and severity of the messages.

Once you specify a filter for the messages, only the messages that fit the filter are displayed on the screen. All the other messages for the enabled tags will get logged into the `ndstrace.log` if the file is set to ON.

Only one filter is applicable at a time. Filter has to be specified for each session of DSTrace.

By default, the severity level is set to INFO, this means that all the messages with severity level more that INFO would be displayed. You can see the severity level by enabling the `svty` tag.

You can use OpenText iMonitor also to filter the trace messages. For more information, refer to "OpenText iMonitor Message Filtering" on page 275.

Linux

Complete the following procedure to filter the trace messages:

1 Enable filtering with the following command:

```
ndstrace tag filter_value
```

To disable filtering, enter the following command:

```
ndstrace tag
```

Examples for enabling filtering:

- To enable filtering for thread ID 35, enter the following:

  ```
  ndstrace thrd 35
  ```

- To enable filtering for severity level fatal, enter the following:

  ```
  ndstrace svty fatal
  ```

  Severity levels can be FATAL, WARN, ERR, INFO, and DEBUG.

- To enable filtering for connection ID 21, enter the following:

  ```
  ndstrace conn 21
  ```

Examples for disabling filtering:

- To disable filtering based on thread ID, enter the following:

  ```
  ndstrace thrd
  ```

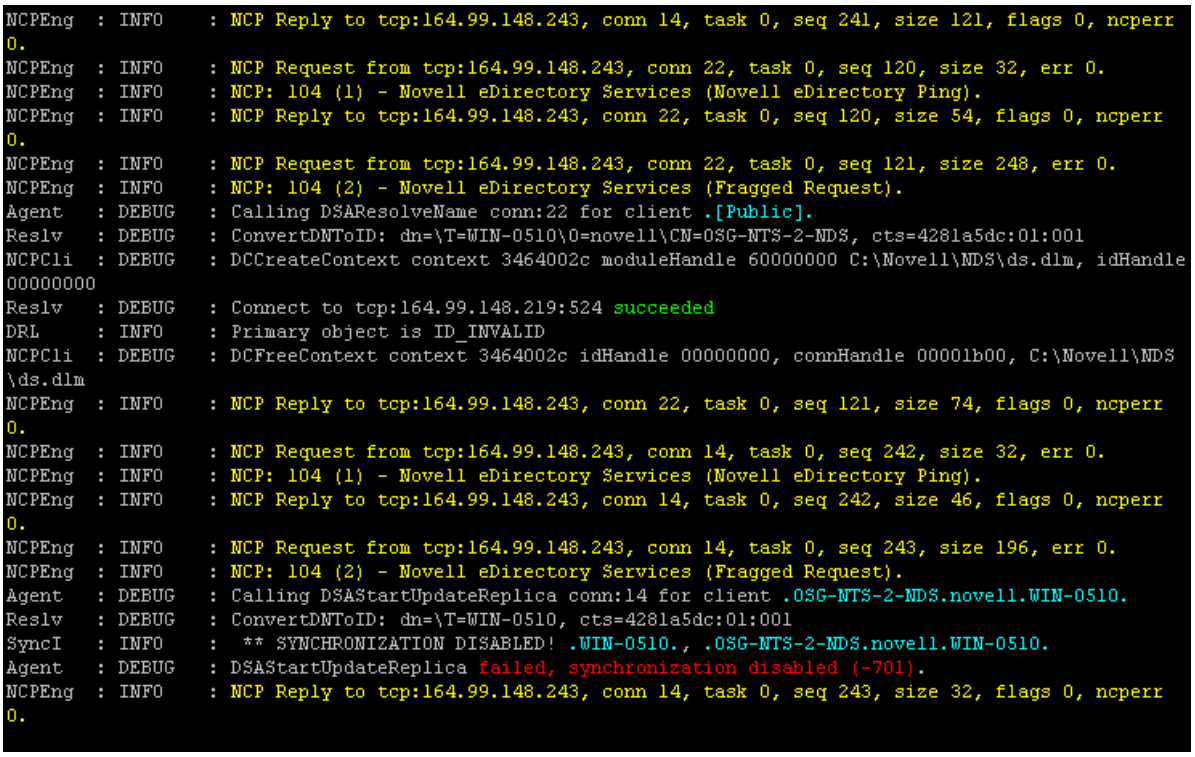
- To disable filtering based on connection ID, enter the following:

  ```
  ndstrace conn
  ```

- To disable filtering based on severity, enter the following:

  ```
  ndstrace svty
  ```

Figure 8-5 *Sample Trace Message Screen With Filters*



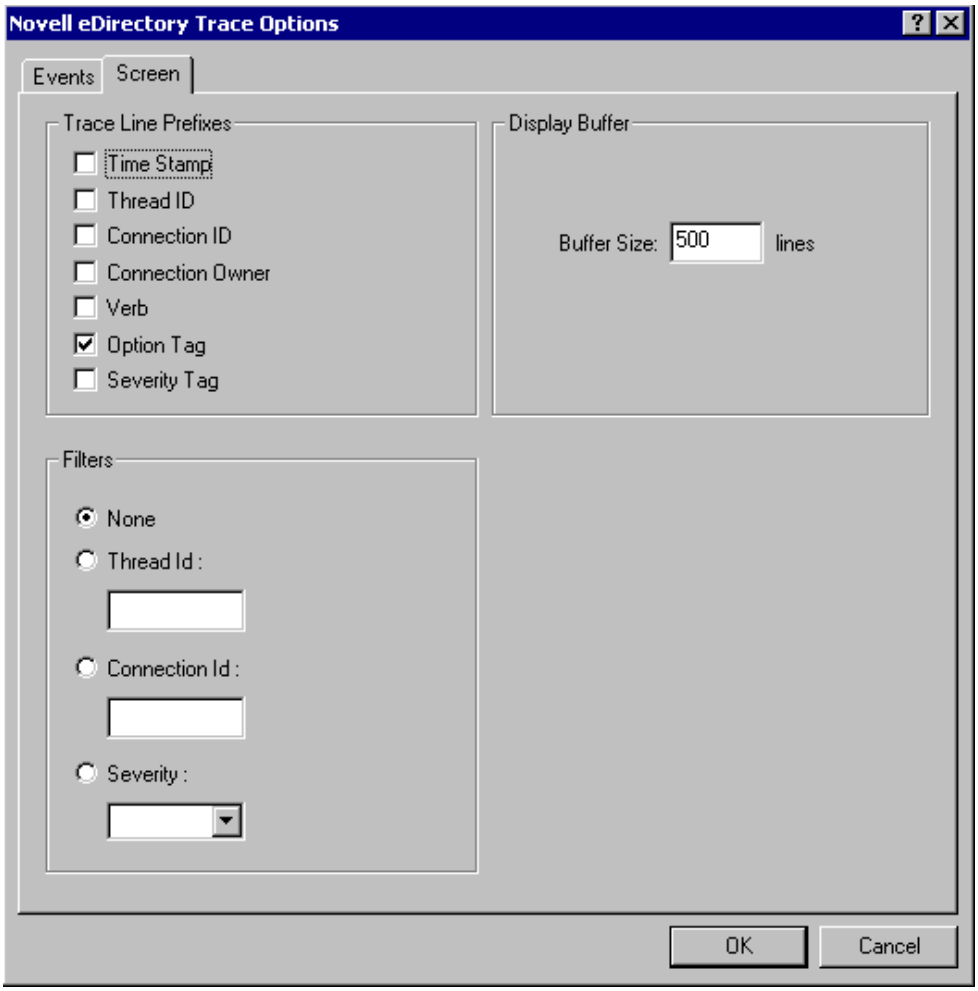
Windows

Complete the following procedure to filter the trace messages:

1. Select **Start** > **Control Panel** > **OpenText eDirectory Services**
2. In the **Services** tab, select **dstrace.dlm**.
3. Click **Edit** > **Options** in the Trace window.

   The OpenText eDirectory Trace Options dialog box is displayed.

Figure 8-6 *Trace Options Screen on Windows*



4 Click on the **Screen** tab.

5 Select the filter option from the **Filters** group and enter the filter value.

You can filter the messages based on:

- Thread ID
- Connection ID
- Severity

Before selecting any of the filters, ensure that it is enabled under **Trace Line Prefixes**.

You can also disable the filtering by selecting **None** or unselecting the filter option.

NOTE: • If you've selected **Thread ID** or **Connection ID** as your filter option and enter a value that does not exist, then the messages won't be displayed on the screen. However, all the other messages will still get logged to the `ndstrace.log` file.

- Trace level severity does not work on Windows.

OpenText iMonitor Message Filtering

You can filter the OpenText iMonitor trace messages based on the connection ID, thread ID, or error number.

To filter based on the connection ID and thread ID, ensure that you have enabled them in the Trace Configuration tab.

For more information, refer to the OpenText iMonitor online help.

SAL Message Filtering

SAL has been enhanced to log extensive information on errors on demand. Function calls can be traced with arguments in the debug builds.

Configuring the Severity Levels

You can use the SAL_LogLevels parameter to configure the severity levels for the SAL messages. SAL_LogLevels is a comma-separated list of desired log levels.

The log levels are explained in the table below:

Table 8-4 *SAL Message Filtering Parameters*

Parameter Name	Description
LogCrit	Critical Messages. This level is enabled by default. After a critical error is logged, the system shuts down.
LogErr	All Error messages. The system continues to function, but the results are unpredictable.
LogWarn	Warning messages. This is just a warning given so that you are aware of some impending error.
LogInfo	Informational messages.
LogDbg	Debug messages used for debugging at the time of development. These messages are compiled out from a release build to reduce the binary size.
LogCall	Traces the function calls. These are subset of Debug messages.
LogAll	Enables all the messages except LogCall.

A “-” at the beginning of a specific log level disables that level.

Examples

To filter based on all the log levels, except LogInfo and LogDbg, complete the following steps:

Linux

1. Stop ndsd.
2. Type the following command:

   ```
   export SAL_LogLevels=LogAll,-LogInfo,-LogDbg
   ```

3. Start ndsd.

Windows

1. Shutdown the DHost.
2. Type the following command at command prompt:

   ```
   set SAL_LogLevels=LogAll,-LogInfo,-LogDbg
   ```

   ```
   c:\novell\nds>dhost.exe /datadir=c:\novell\nds\DIBFiles\
   ```

3. Restart DHost.

Setting the Log File Path

You can use the SAL_LogFile environment variable to specify the log file location. This can be a valid file name with a valid path, or one of the following.

- Console: All messages are logged to the console.
- Syslog: In Linux, the messages go to the syslog. On Windows, messages are logged into a file with the name syslog. This is the default behavior for logging.

  All critical errors are always logged to syslog unless it is disabled specifically.

9 SecretStore Configuration for OpenText eDirectory Server

NOTE: We will be deprecating support for Secret Store post OpenText eDirectory 9.2.7. There will be no support provided for issues related to the secret store post OpenText eDirectory 9.2.7 and above.

SecretStore executables and libraries are installed by default with OpenText eDirectory installation. However, SecretStore configuration is optional for a new installation of OpenText eDirectory. For OpenText eDirectory server upgrade, no changes are made to the existing configuration. Ensure you extend the OpenText eDirectory schema for SecretStore functionality on Linux and Windows platforms using the following command:

```
ice -S SCH -f /var/opt/novell/eDirectory/lib/nds-schema/sssv3.sch -D LDAP -
s
<serverIP> -d <adminDN>
```

For example,`ice -S SCH -f /var/opt/novell/eDirectory/lib/nds-schema/sssv3.sch -D LDAP -s 1.2.3.4 -d cn=admin,o=administrators`

Use the procedures given in the following sections to configure and deconfigure SecretStore:

- "Linux" on page 277
- "Windows" on page 278

Linux

Configuring SecretStore

Use the following steps to configure the SecretStore:

1. To configure, run `ssscfg -c`.
2. Add an entry `ssncp` in the `/etc/opt/novell/eDirectory/conf/ndsmodules.conf` to load SecretStore module by default while OpenText eDirectory is being started. You can also use `nss` utility to load or unload the SecretStore module later.

Deconfiguring SecretStore

For deconfiguration, run the `ssscfg -d` command. Remove the `ssncp` entry if it exists in the `/etc/opt/novell/eDirectory/conf/ndsmodules.conf` location.

Windows

Use the following steps to configure and deconfigure the SecretStore:

1. For configuration, run `ssscfg.exe -c.`
2. For deconfiguration, run `ssscfg.exe -d.`

The `ssscfg.exe` utility exists in the `eDirectoryInstallDrive:>\Novell\NDS\` directory. To autoload the SecretStore module during OpenText eDirectory server startup, set the `ssncp.dlm` module to `auto` from the GUI interface of the `NDSCons.exe`.

10 Merging OpenText eDirectory Trees

The OpenText eDirectory Merge utility allows you to merge two separate OpenText eDirectory trees into a single OpenText eDirectory tree. Only the Tree objects are merged. Container objects and their leaf objects maintain separate identities within the newly merged tree.

TIP: To move leaf objects or merge partitions, use OpenText Identity Console.

The two trees you merge are called the local source tree and the target tree. Before merging one tree into another tree, the target tree should have all but one replica of the root partition removed. When there is only one replica of the root partition in the target tree, you can proceed with the merge. After the merge, there will be two replicas of the root partition-the replica that was on the target tree and the replica that was on the source tree server that ran the merge operation. If you need additional replicas of the root partition in your tree, you can place them after the merge has completed.

If the target tree server contains more than one replica of the root partition when the merge takes place, servers not holding the master replica might have a problem with the placement of external reference objects. These objects are contained in subordinate reference partition roots that must be placed on the other servers that have a replica of the root partition to represent partition boundaries. For each partition subordinate to the root partition in the source tree, there must be a subordinate reference partition root placed in the target tree. If there is a failure, it will report an OpenText eDirectory error code of -605 for synchronization status. In this case, use DSRepair to run a local database repair on the server producing the error. See “Performing a Local Database Repair” on page 316 for more information.

DSMerge does not change OpenText eDirectory names or contexts within the containers. Object and property rights for the merged objects are retained.

This chapter contains the following topics:

- “Merging OpenText eDirectory Trees” on page 279
- “Grafting a Single Server Tree” on page 285
- “Renaming a Tree” on page 290
- “Using the Client to Merge Trees” on page 291

Merging OpenText eDirectory Trees

To merge OpenText eDirectory trees, use the Merge Tree command. This command lets you merge the root of two separate OpenText eDirectory trees. Only the Tree objects are merged. Container objects and their leaf objects maintain separate identities within the newly merged tree.

The two trees you merge are called the source tree and the target tree. The target tree is the tree that the source tree will be merged into.

DSMerge does not change object names within the containers. Object and property rights for the merged tree are retained.

- "Prerequisites" on page 280
- "Target Tree Requirements" on page 280
- "Schema Requirements" on page 280
- "Merging the Source into the Target Tree" on page 281
- "Partition Changes" on page 281
- "Preparing the Source and Target Trees" on page 282
- "Synchronizing Time before the Merge" on page 282
- "Merging Two Trees" on page 283
- "Post-Merge Tasks" on page 284

Prerequisites

- ❒ OpenText eDirectory must be installed on the server containing the master replica of the source tree's [Root] partition.
- ❒ Other servers in the source tree should be upgraded to OpenText eDirectory 8.8 or later to ensure proper functionality.

NOTE: To delete Authorized Login Methods, use the ldapdelete tool or OpenText Identity Console.

Target Tree Requirements

- ❒ OpenText eDirectory must be installed on the server containing the master replica of the target tree's [Root] partition. If this server is running any other version of NDS® or OpenText eDirectory, the merge operation will not complete successfully.
- ❒ Other servers in the target tree should be upgraded to OpenText eDirectory 8.8 or later to ensure proper functionality.
- ❒ You cannot maintain containers with the same name subordinate to Tree in both the source and target trees. Before merging two trees, one of the containers must be renamed.
- ❒ If both the source and target trees have a Security object, one of them must be removed before merging the trees.

Schema Requirements

Before attempting to perform a merge operation, the schema of both trees must match exactly. You should run DSRepair on the server containing the master replica of the [Root] partition for each tree. Use the Import Remote Schema option to ensure that each tree is aware of all schema in the other tree.

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.

3 Choose option 4 Import schema from Tree (Global Schema Options).

4 Provide the IP address or tree name.

Merging the Source into the Target Tree

When you merge the trees, the servers in the source tree become part of the target tree.

The target Tree object becomes the new Tree object for objects in the source tree, and the tree name of all servers in the source tree is changed to the target tree's name.

After the merge, the tree name for the target tree servers is retained.

The objects that were subordinate to the source Tree object become subordinate to the target Tree object.

Partition Changes

During the merge, DSMerge splits the objects below the source Tree object into separate partitions.

All replicas of the Tree partition are then removed from servers in the source tree, except for the master replica. The server that contained the master replica of the source tree receives a replica of the target tree's Tree partition.

Figure 10-1 and Figure 10-2 illustrate the effect on partitions when you merge two trees.

Figure 10-1 *OpenText eDirectory Trees before a Merge*

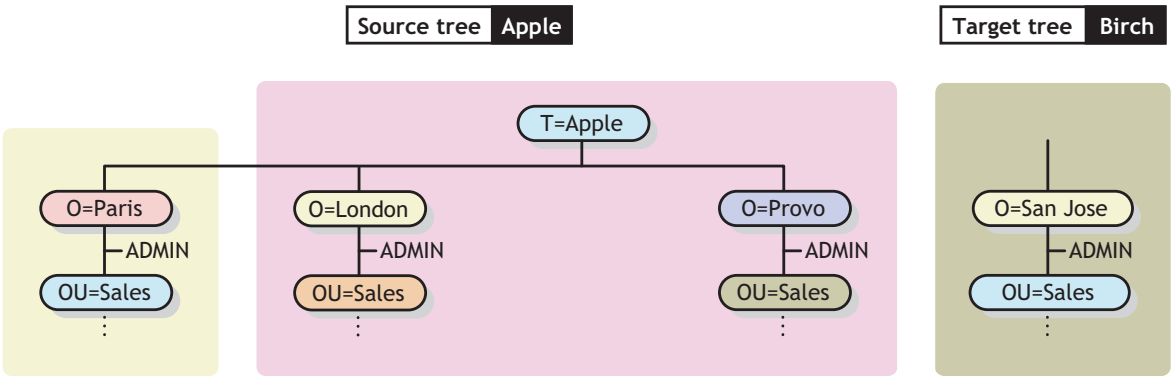
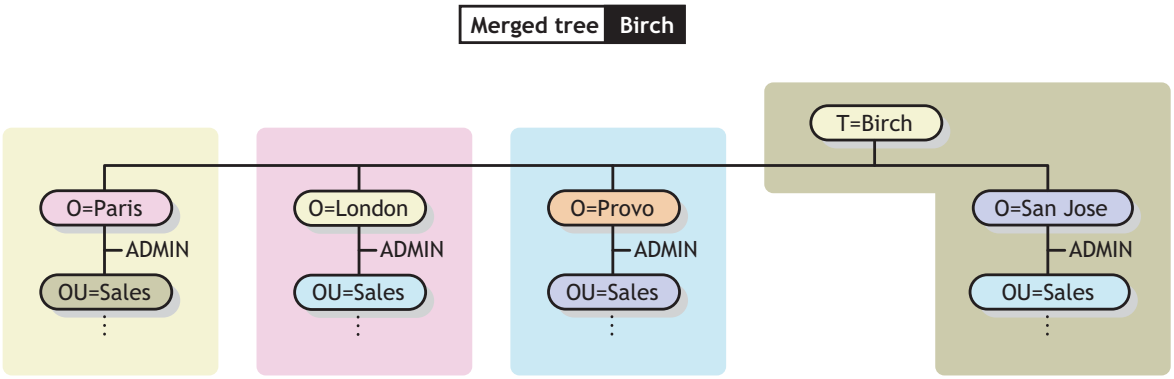


Figure 10-2 *Merged OpenText eDirectory Tree*



Preparing the Source and Target Trees

Before performing a merge operation, ensure that the state of synchronization for all servers affected by the operation is stable. The following table provides prerequisites for preparing source and target trees for merging.

Prerequisite	Required Action
WANMAN should be turned off on all servers that hold a replica of the source tree's Tree partition or the target tree's Tree partition.	Review your WANMAN policy so that WAN communication restrictions do not interfere with the merge operation. If required, turn WANMAN off before initiating the merge operation.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.
No identical names can exist between the source and target trees.	Rename objects on the source and target trees if identical names exist. Move objects from one of the containers to a different container in its tree if you don't want to rename the container objects, then delete the empty container before running DSMerge. You can have identical container objects in both trees if they are not immediately subordinate to the Tree object.
No login connections should exist on the source tree.	Close all connections on the source tree.
The OpenText eDirectory version must be the same on both the source and target trees.	Upgrade all OpenText non-eDirectory servers that have a replica of the root partition.
The target tree must have only one copy of the root replica.	Remove all replicas on the target tree except the master replica.
The schema on both the source and target trees must be the same.	Run DSMerge. If reports indicate schema problems, use DSRepair to match the schemas. See “Importing Remote Schema” on page 324 for more information. Run DSMerge again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target trees have a security container, remove one container as explained in Appendix A, “NMAS Considerations,” on page 759.

Because the merge operation is one single transaction, it is not subject to catastrophic failure caused by power outages or hardware failure. However, you should perform a regular backup of the eDirectory database before using DSMerge. For more information, see Chapter 15, “Backing Up and Restoring OpenText eDirectory,” on page 417.

Synchronizing Time before the Merge

IMPORTANT: Proper configuration of time synchronization is a very involved process. Make sure you allow enough time to synchronize both trees before you merge the trees.

OpenText eDirectory will not work properly if different time sources are used that have different times or if all servers in a tree are not time synchronized.

Before you do the merge, make sure that all servers in both trees are time synchronized and that they use only one time server as a time source. However, the target tree time can be ahead of the source tree time by as much as five minutes.

Generally, there should be only one Reference or one Single time server in a tree. Likewise, after the merge, the tree should contain only one Reference or one Single time server.

If each of the trees you are merging has either a Reference or a Single time server, reassign one of them to refer to the Reference or Single time server in the other tree so that the final merged tree contains only one Reference or Single time server.

For more information on time server types, see "Time Services" in the *OES Planning and Implementation Guide* (http://www.novell.com/documentation/oes11/oes_implement_lx/data/time.html).

Merging Two Trees

For complete functionality of all menu options, run DSMerge on a server that contains the master replica of the Tree partition.

If you don't know where the master replica is stored, you will be prompted with the correct server name when you attempt an operation that requires the master replica.

To perform a merge operation, use the following method:

- The command line client

  For more information, see "Using the Client to Merge Trees" on page 291.

When merging large trees, it is significantly faster to designate the tree with the fewest objects immediately subordinate to the Tree object as the source tree. By doing this, you create fewer partition splits during the merge, because all objects subordinate to the Tree object result in new partitions.

Because the source tree name no longer exists after the merge, you might need to change your client workstation configurations. For the Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the `net.cfg` files. For the Novell Client for Windows, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

TIP: To minimize the number of client workstations you need to update, designate the tree with the most client workstations as the target tree, because the final tree retains the name of the target tree. Or rename the tree after the merge operation so that the final tree name corresponds to the tree with the greater number of client workstations attaching to it. For more information, see "Renaming a Tree" on page 290.

Use the following list of prerequisites to determine readiness for the merge operation:

- ❒ You have access to the source tree server through OpenText Identity Console
- ❒ You have the name and password of the Administrator objects that have Supervisor object rights to the Tree object of both trees you want to merge
- ❒ The OpenText eDirectory database for the two trees has been backed up
- ❒ All servers in both trees are synchronized and using the same time source
- ❒ (Optional) All servers in the tree are operational (Servers that are down will update automatically when they are operational.)
- ❒ Review the merge prerequisites listed in "Preparing the Source and Target Trees" on page 282

The merge process takes a few minutes, but there are other variables that increase the length of time for the merge operation to complete:

- Many objects subordinate to the Tree object that must be split into partitions
- Many servers in the source tree that require a tree name change

To merge two trees follow the procedure as explained in the section: Using the DSMerge eMTool.

NOTE: Do not merge two partitions when the parent partition is not EBA–enabled and the child partition is EBA–enabled. Doing this might break the EBA functionality.

Post-Merge Tasks

Following the merging of two trees, it might be necessary to complete the following steps:

1. Verify that all tree names were changed correctly.
2. Check the new partitions that the merge operation created.

   If you have many small partitions in the new tree, or if you have partitions that contain related information, you might want to merge them. For more information, see "Merging a Partition" on page 144.
3. Re-create any leaf objects or aliases in the tree that were deleted before you ran DSMerge.
4. Evaluate partitioning of the OpenText eDirectory tree.

   Merging trees might change replica placement requirements on the new tree. You should carefully evaluate and change the partitioning as needed.
5. Update your client workstation configuration.

   For the Novell Client for Windows, check the Preferred Tree and Preferred Server statements on the client Property Page, or rename the target tree.

   If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

The Access Control List (ACL) for the Tree object of the source tree is preserved. Therefore, the rights of the source tree's user Admin to the Tree object are still valid.

After the merge is complete, both admin users still exist and are uniquely identified by different container objects.

For security reasons, you might want to delete one of the two Admin User objects or restrict the rights of the two objects.

Grafting a Single Server Tree

The **Graft Tree** option lets you graft a single server source tree's Tree object under a container specified in the target tree. After the graft is completed, the source tree receives the target tree's name.

During the graft, DSMerge changes the object class of the source tree's Tree object to Domain and makes a new partition. The new Domain object is the partition root for the new partition. All the objects under the source tree's Tree object are located under the Domain object.

The target tree's administrator has rights to the resulting tree's root container and, therefore, has rights to the source tree's grafted root.

NOTE: It might take up to several hours for the inherited rights to be recalculated and become effective. This time will vary based on the tree's complexity, size, and number of partitions.

The source tree's administrator has rights only in the newly created Domain object.

Figure 10-3 and Figure 10-4 illustrate the effects of grafting a tree into a specific container.

Figure 10-3 *OpenText eDirectory Trees before a Graft*

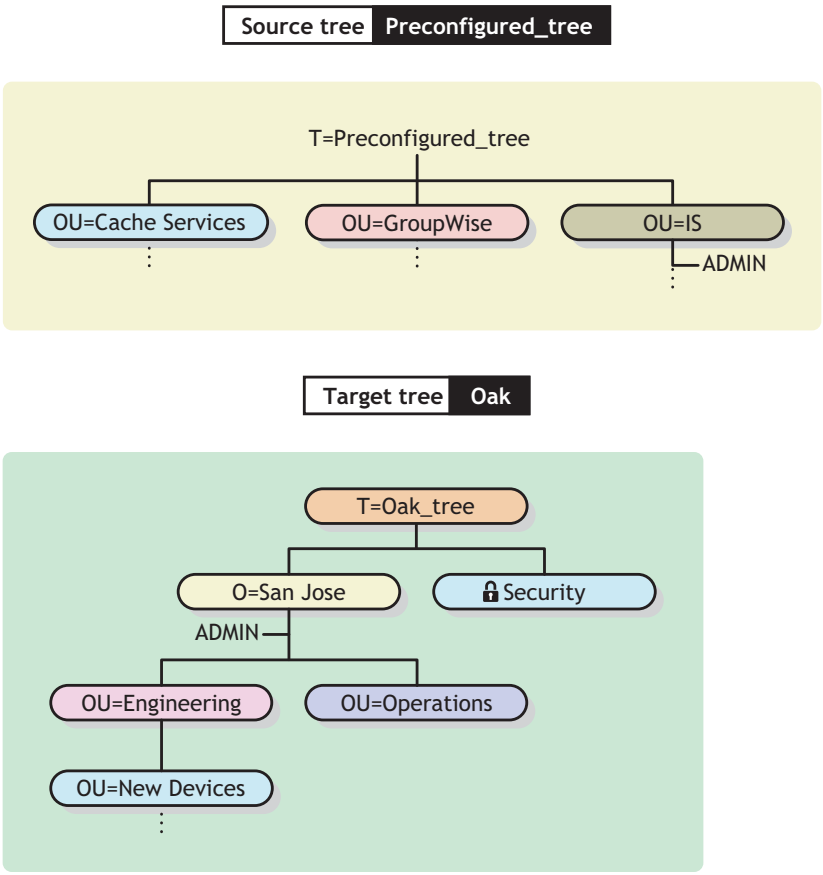
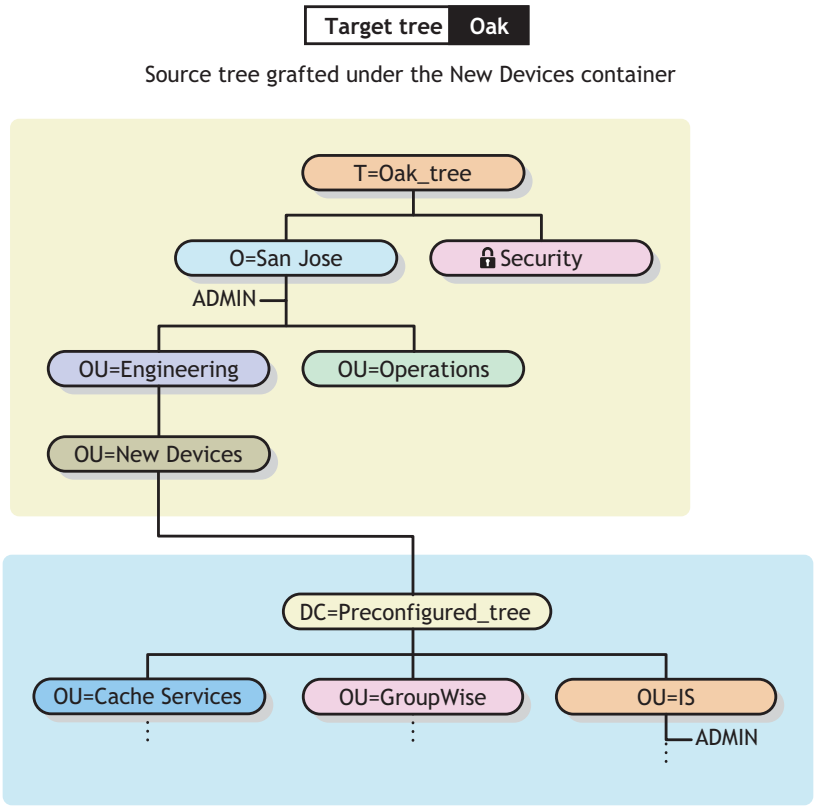


Figure 10-4 *Grafted OpenText eDirectory Tree*



This sections contains the following information:

- "Understanding Context Name Changes" on page 287
- "Preparing the Source and Target Trees" on page 288
- "Containment Requirements for Grafting" on page 289
- "Grafting the Source and Target Tree" on page 290

Understanding Context Name Changes

After the source tree has been grafted into the target tree container, the distinguished names for objects in the source tree will be appended with the source tree's name followed by the distinguished name of the target tree's container name where the source tree was merged. The relative distinguished name will remain the same.

For example, if you are using dot delimiters, the typeful name for Admin in the Preconfigured_tree (source tree) is

```
CN=Admin.OU=IS.T=Preconfigured_tree
```

After the Preconfigured_tree is merged into the New Devices container in the Oak_tree, the typeful name for Admin is -

```
CN=Admin.OU=IS.DC=Preconfigured_tree.OU=Newdevices.
OU=Engineering.O=Sanjose.T=Oak_tree.
```

NOTE: The maximum number of characters allowed in a DN of any type, including a container DN, is 255 characters. This limitation is particularly important when you are grafting the root of one tree into a container near the bottom of the target tree.

The last dot following Oak_tree (`Oak_tree.`) indicates that the last element in the distinguished name is the tree name. If you leave off the trailing dot, then also leave off the tree name.

Preparing the Source and Target Trees

Before initiating the graft operation, ensure that the state of all of the servers affected by the operation is stable. The following table provides prerequisites for preparing the source and target trees before grafting.

Prerequisite	Required Action
The source tree must have only one server.	Remove all but one server from the source tree.
No aliases or leaf objects can exist at the source tree's Tree object.	Delete any aliases or leaf objects at the source tree's Tree object.
No similar names can exist in the graft container.	Rename objects under the target tree graft container or rename the source tree. Move objects from one of the containers to a different container in its tree if you don't want to rename objects, then delete the empty container before running DSMerge. For more information, see Chapter 3, “Managing Objects,” on page 95. You can have identical container objects in both trees if they are not immediately subordinate to the same parent object. Objects are uniquely identified by their immediate container object.
The eDirectory version for both the source tree and target tree container must be 8.51 SP2a or later.	DSMerge will search for the appropriate version of OpenText eDirectory. If an acceptable version isn't found, DSMerge will return an error. You can get the latest version of OpenText eDirectory from the Software License and Download portal (https://www.microfocus.com/en-us/products).
The container where you will join the target tree is in a partition that has no replicas (a single-server partition).	If the target container has multiple replicas, do one of the following: ◆ Make the partition associated with this container the master replica and delete other replicas. ◆ Split the target tree graft container into a separate partition and remove replicas. After the graft is complete, the partition association can be re-established.

Prerequisite	Required Action
The server holding the target container must also hold a replica of the ROOT partition.	If the server doesn't hold a replica of ROOT, the graft will fail and you will see error `-672 No Access` because the directory is unable to verify administrator rights for the target tree. Use OpenText Identity Console to add a replica for ROOT. For more information, see "Adding a Replica" on page 147.
The schema on both the source and target trees must be the same.	Run the Graft option in DSMerge. If reports indicate schema problems, run DSRepair on the target tree to import the schema from the source tree. The graft operation automatically imports the schema from the target tree to the source tree. Run DSMerge again.
Only one tree can have a security container subordinate to the tree root.	If both the source and target tree have the security container, remove one container as explained in Appendix A, "NMAS Considerations," on page 759.
The source tree's time reference must be reconfigured.	The source tree should usually be set as a secondary server configured to get its time source from a server in the target tree. To reconfigure Timesync, see "Configuring and Administering Time Synchronization" (http://www.novell.com/documentation/oes11/oes_implement_lx/data/time.html#time-cfgnadmin) in the *OES Planning and Implementation Guide*.

Containment Requirements for Grafting

To graft a source tree into a target tree container requires that the target tree container be prepared to accept the source tree. The target tree container must be able to contain an object of the class domain. If there is a problem with containment, error `-611 Illegal Containment` will occur during the graft operation.

Use the information in the following table to determine if you need to run DSRepair to modify containment lists.

Target Tree Container Requirements	The target tree container object must include the domain object in its containment list. You can check this using **OpenText iMonitor** > **Schema**. If the containment list does not include Domain, run DSRepair to make schema enhancements.
Source Tree Requirements	The graft operation changes the source tree root from the class Tree Root to the class Domain. All of the object classes that are subordinate to the Tree must be able to be contained by the class Domain according to the schema rules. You can check this using **OpenText iMonitor** > **Schema**. If the containment list does not include Domain, run DSRepair to make schema enhancements.

If containment requirements aren't met, run DSRepair to correct the schema as explained at the section: Running DSRepair on the OpenText eDirectory Server.

Grafting the Source and Target Tree

After you ensure that prerequisites are met, use DSMerge to perform the graft. For more information see: DSMerge eMTool Options.

Renaming a Tree

You must rename a tree if the two trees you want to merge have the same name.

You can rename only the source tree. To rename the target tree, run the Rename Tree Wizard in OpenText Identity Console against a server on the target tree.

If you change a tree name, the bindery context does not automatically change. Because the bindery context set in the `autoexec.ncf` file also contains the tree name (for example, `SET Bindery Context = O=n.`*`test_tree_name`*), a server with a recently changed tree name does not use the context that it used before the tree name change.

Therefore, after you change a tree's name, you might need to change your client workstation configurations. For the Novell Client for DOS/Windows, check the Preferred Tree and Preferred Server statements in the `net.cfg` files. For Novell Client for Windows, check the Preferred Tree and Preferred Server statements on the client Property Page.

If Preferred Server is used, the client is unaffected by a tree merge or rename operation because the client still logs in to the server by name. If Preferred Tree is used and the tree is renamed or merged, then that tree name no longer exists. Only the target tree name is retained after the merge. Change the preferred tree name to the new tree name.

When you merge two trees, to minimize the number of client workstations that need to be updated, designate the tree with the most client workstations as the target tree because the final tree retains the name of the target tree.

You can also rename the tree after the merge so that the final tree name corresponds to the tree name with the majority of client workstations.

Another option is to rename the merged tree to the name of the original source tree. If you choose this option, then you must update the `net.cfg` files on the target tree client workstations.

Use the following list of prerequisites to determine readiness for the renaming operation:

- ❒ Access to a server console on the source tree or an established RCONSOLE session with the server
- ❒ The Supervisor object right to the Tree object of the source tree
- ❒ (Optional) All servers in the tree are operational (Servers that are down will update automatically when they are operational.)

To rename the tree, follow the information as explained at: DSMerge eMTool Options.

Using the Client to Merge Trees

The OpenText eDirectory Management Toolbox (eMBox) Client is a command line Java client that gives you remote access to DSMerge. The `emboxclient.jar` file is installed on your server as part of eDirectory. You can run it on any machine with a JVM. For more information on the Client, see "Using the Command Line Client" on page 560.

Using the DSMerge eMTool

1. Run the Client in interactive mode by entering the following at the command line:

   `java -cp` *`path_to_the_file/`*`emboxclient.jar -i`

   (If you have already put the `emboxclient.jar` file in your class path, you need to enter only `java -i`.)

   The Client prompt appears:

   ```
   Client>
   ```

2. Log in to the server that will run DSMerge (this will be the source tree) by entering the following:

   `login -s`*`server_name_or_IP_address`* `-p`*`port_number`*
   `-u`*`username.context`* `-w`*`password`* `-n`

   The port number is usually 80 or 8028, unless you have a Web server that is already using the port. The `-n` option opens a nonsecure connection.

   The Client will indicate whether the login is successful.

3. Enter a merge command, using the following syntax:

   `dsmerge.`*`task options`*

   For example, `dsmerge.m -uadmin -ptest -TApple -Uadmin -Ptest` merges the target tree `Apple` (with target tree user name `Admin` and user password `test`) with the source tree you are currently logged in to (with source tree user name `Admin` and user password `test`).

   `dsmerge.g -uadmin -ptest -TOrange -Uadmin -Ptest -CFruit` grafts the source tree you are currently logged in to (with source tree user name `Admin` and user password `test`) into the `Fruit` container in the target tree `Orange` (with target tree user name `Admin` and user password `test`).

   A space must be between each switch. The order of the switches is not important.

   The Client will indicate whether the DSMerge operation was successful.

   See "DSMerge eMTool Options" on page 292 for more information on the DSMerge eMTool options.

4. Log out from the Client by entering the following command:

   `logout`

5. Exit the Client by entering the following command:

   `exit`

DSMerge eMTool Options

The following tables lists the DSMerge eMTool options. You can also use the `list -t dsmerge` command in the Client to list the DSMerge options with details. See "Listing eMTools and Their Services" on page 563 for more information.

Merge Operation	Client Command
Check whether the tree can be renamed	`dsmerge.pr -u`*`User`* `-p`*`User_password`* `-n`*`New_tree_name`*
Rename the tree	`dsmerge.r -u`*`User`* `-p`*`User_password`* `-n`*`New_tree_name`*
Check whether two trees can be merged	`dsmerge.pm -u`*`Source_tree_user`*`-p`*`Source_tree_user_password`* `-T`*`Target_tree_name`* `-U`*`Target_tree_user`* `-P`*`Target_tree_password`*
Merge two trees	`dsmerge.m -u`*`Source_tree_user`*`-p`*`Source_tree_user_password`* `-T`*`Target_tree_name`*`-U`*`Target_tree_user`* `-P`*`Target_tree_password`*
Check whether the source tree can be grafted into the target tree container	`dsmerge.pg -u`*`Source_tree_user`* `-p`*`Source_tree_user_password`* `-T`*`Target_tree_name`* `-U`*`Target_tree_user`* `-P`*`Target_tree_password`* `-C`*`Target_tree_container`*
Graft the source tree into the container in the target tree	`dsmerge.g -u`*`Source_tree_user`* `-p`*`Source_tree_user_password`* `-T`*`Target_tree_name`* `-U`*`Target_tree_user`* `-P`*`Target_tree_password`* `-C`*`Target_tree_container`*
Cancel the running DSMerge operation	`cancel`

11 Encrypting Data in OpenText eDirectory

OpenText eDirectory lets you encrypt specific data when the data is:

- Stored on the disk.
- Transmitted between two or more OpenText eDirectory servers. This provides greater security for the confidential data.

You can protect data by encrypting the following:

- Attributes: For protecting confidential data stored on the disk.

  See "Encrypted Attributes" on page 293.
- Replication: For protecting confidential data during replication between OpenText eDirectory servers.

  See "Encrypted Replication" on page 303.

IMPORTANT: With the introduction of Enhanced Background Authentication (EBA), the data is automatically encrypted during data replication between EBA-enabled OpenText eDirectory servers. If one of the servers is not EBA-enabled, you can configure encrypted replication policies to encrypt the data.

Encrypted Attributes

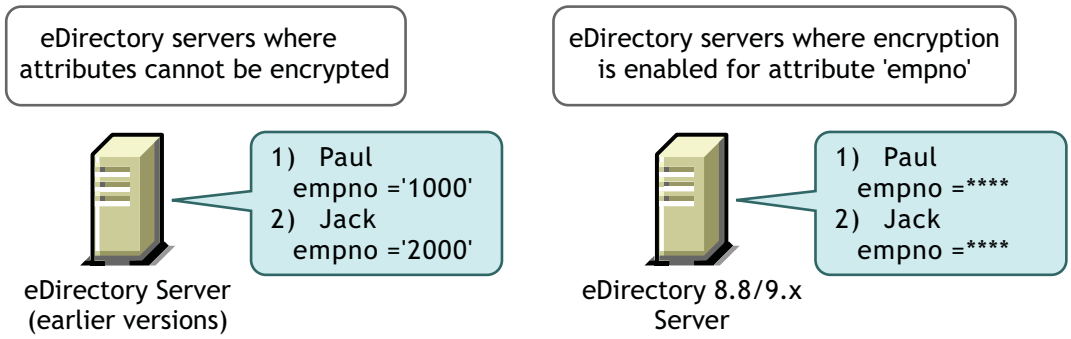
You can encrypt the attributes to protect data while they are stored on the disk. Encrypted attributes is a server-specific feature. You can use this feature in scenarios where you need to protect confidential data such as credit card numbers of bank customers.

When you encrypt an attribute, the value of the attribute is encoded. For example, you can encrypt an attribute `empno` stored in the DIB. If empno=1000, then the value of the attribute (1000), is not stored as clear text on the disk. You can read this encrypted value only when you access the directory over a secure channel.

All attributes in a schema can be enabled for encryption. However, we recommend you not to enable Common Name (CN) attribute for encryption and enable only the sensitive data for encryption. Refer to "Achieving Complete Security While Encrypting Data" on page 310 before you decide on marking any attributes for encryption.

There is no limitation in accessing Public and Server readable encrypted attributes, this means that a client can access these attributes over clear text but you can mark these attributes for encryption at the DIB level. Enabling encryption on an attribute which is flagged `[Public Read]` in schema, does not prevent it from being accessed via non-secure methods.

Figure 11-1 *Encrypted Attributes*



The data in OpenText eDirectory can be stored in any of the following ways:

- In the Data Information Base (DIB) or database
- As backup data
- LDIF file

You can encrypt attributes by creating and applying encrypted attributes policies to the servers.

To encrypt the attributes, do the following using OpenText Identity Console:

1. Create and define an encrypted attribute policy.
   - **1a** Select the attributes for encryption.
   - **1b** Select the encryption scheme for the attributes.

     Refer to "Creating Encrypted Attributes Policies" on page 297 for more information.
2. Apply the encrypted attributes policy to a server.

   Refer to "Editing / Modifying the Encrypted Attributes Policies" on page 297 for more information.

You can also encrypt attributes through LDAP.

Refer to "Managing Encrypted Attributes Policies Through LDAP" on page 297 for more information.

NOTE: Encrypted Attributes Policy assignment takes effect when Limber runs.

As a best practice, we recommend you to do the following:

- Mark only sensitive attributes for encryption. Do not mark all attributes for encryption (for example, public or server readable attributes).
- Use AES while marking an attribute for encryption as it is a strong encryption algorithm.

The rest of this section provides the following information:

- "Using Encryption Schemes" on page 295
- "Managing Encrypted Attributes Policies" on page 295
- "Accessing the Encrypted Attributes" on page 300
- "Viewing the Encrypted Attributes" on page 301

- “Encrypting and Decrypting Backup Data” on page 302
- “Cloning the DIB Fileset Containing Encrypted Attributes” on page 302
- “Adding OpenText eDirectory Servers to Replica Rings” on page 302
- “Backward Compatibility” on page 302
- “Migrating to Encrypted Attributes” on page 303
- “Replicating the Encrypted Attributes” on page 303

Using Encryption Schemes

OpenText eDirectory provides the highest level of security for an attribute by supporting the following encryption schemes:

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

You can select different encryption schemes for different attributes in a single encrypted attributes policy. For example, in an encrypted attributes policy EP1, you can select both AES as the encryption scheme for an attribute cubeno and Triple DES for an attribute empno. Refer to “Creating Encrypted Attributes Policies” on page 297 for more information.

You can change the encryption scheme for an encrypted attribute by editing the encrypted attributes policy. You can also decrypt an attribute that you have encrypted earlier. Refer to “Editing / Modifying the Encrypted Attributes Policies” on page 297 for more information.

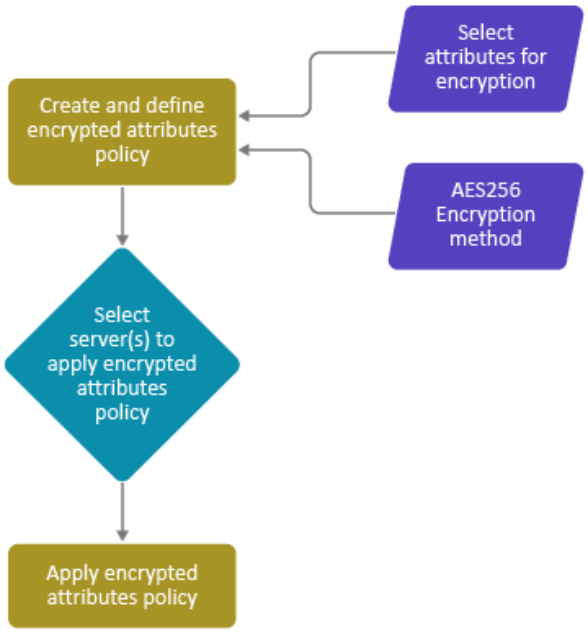
You can choose to have different encryption schemes in different servers of the replica ring. For example, an attribute might be enabled for encryption using AES on Server1, Triple DES on Server2 and no encryption scheme on Server3.

Managing Encrypted Attributes Policies

You can manage encryption of the attributes by creating and defining policies and applying them to servers.

You define an encrypted attributes policy by selecting the attributes for encryption and an encryption scheme.

Figure 11-2 *Encrypting Attributes*



You can manage encrypted attributes policies using OpenText Identity Console. This section provides the following information:

- "Managing Encrypted Attributes Policies Through OpenText Identity Console" on page 296
- "Managing Encrypted Attributes Policies Through LDAP" on page 297
- "Copying the Encrypted Attributes Policies" on page 299
- "Partition Operations" on page 299

Managing Encrypted Attributes Policies Through OpenText Identity Console

This section contains the following procedures:

- "Creating Encrypted Attributes Policies" on page 297
- "Editing / Modifying the Encrypted Attributes Policies" on page 297
- "Deleting Encrypted Attributes Policies" on page 297

If encrypted attributes are present in the OpenText eDirectory server, OpenText Identity Console behaves in the following manner:

1. Reading, listing, or modifying encrypted attributes is not allowed over clear text or secure channel.
2. An entry that has non-encrypted attributes is not allowed to read, list, or modify attributes through OpenText Identity Console over clear text or secure channel. This implies that the whole entry is blocked.

Creating Encrypted Attributes Policies

1. On the OpenText Identity Console home page, click **Encrypted Attributes** tile.
2. On the **Encrypted Attributes** page, click **Create EA Policy** +.
3. Enter policy name at **Name** field.
4. Click Context and select the container.
5. Click NCP Server and select the server > click **OK**.
6. Select the attribute. The selected attributes are displayed under **Selected Attribute** table. >
7. Click **Create**.

   The **Policy created successfully** message appears.

Editing / Modifying the Encrypted Attributes Policies

1. On the OpenText Identity Console home page, click **Encrypted Attributes** tile.
2. In the **Encrypted Attributes** page, select the policy that must be modified, and click Modify Object .
3. Modify the required information in the **Modify Encrypted Attributes Policy** Management page to edit the policy.
4. Click **Finish**.

   The **Policy Modified Successfully** message appears.
5. Click **OK**.

Deleting Encrypted Attributes Policies

1. On the OpenText Identity Console home page, click **Encrypted Attributes** tile.
2. In the **Encrypted Attributes** page, select the policy that need must be deleted, and click Delete Object . Click **OK** to confirm the deletion of selected attribute.

Managing Encrypted Attributes Policies Through LDAP

IMPORTANT: We strongly recommend that you use OpenText Identity Console for managing encrypted attributes and not LDAP.

This section contains the following procedures:

- "Creating and Defining Encrypted Attributes Policies" on page 298
- "Editing Encrypted Attributes Policies" on page 298
- "Applying Encrypted Attributes Policy" on page 299
- "Deleting Encrypted Attributes Policy" on page 299

NOTE: You should specify the attribute and scheme pair while marking any attribute through LDIF for encryption and not the list of attributes and scheme. This is the current limitation with encrypted attributes.

Creating and Defining Encrypted Attributes Policies

1. Create an attribute encryption policy.

   For example, the encrypted attributes policy is AE Policy- test-server, then

   ```
   dn: cn=AE Policy - test-server, o=novell
   changetype: add
   objectClass: encryptionPolicy
   ```

2. Add the attrEncryptionDefinition attribute to the Policy object you created and mark the attributes for encryption.

   For example, if the attribute name you want to encrypt is CRID then specify the encryption scheme and attribute name as mentioned below:

   ```
   dn: cn=AE Policy - test-server, o=novell
   changetype: modify
   add: attrEncryptionDefinition
   attrEncryptionDefinition: aes$CRID
   ```

   NOTE: Attribute name specifies the NDS name for the attribute. Many attributes in OpenText eDirectory have both an LDAP name and an NDS name. Here, specify the attribute name that requires the NDS name.

   ---

3. Add the attrEncryptionRequiresSecure attribute to the policy.

   The value of this attribute specifies whether a secure channel is always necessary to access the encrypted attributes. The value 0 means that it is not always necessary. The value 1 means that it is always necessary.

   For example:

   ```
   dn: cn=AE Policy - test-server, o=novell
   changetype: modify
   add: attrEncryptionRequiresSecure
   attrEncryptionRequiresSecure: 0
   ```

4. Associate the policy with an NCP server.

   For example, if the NCP server is test-server:

   ```
   dn: cn=test-server, o=novell
   changetype: modify
   add: encryptionPolicyDN
   encryptionPolicyDN: cn=AE Policy - test-server, o=novell
   ```

Editing Encrypted Attributes Policies

The following LDIF file illustrates editing an encrypted attributes policy by changing the value of the attrEncryptionRequireSecure attribute:

```
dn: cn=AE Policy - test-server, o=novell
changetype: modify
replace: attrEncryptionRequiresSecure
attrEncrytionRequiresSecure: 1
```

Applying Encrypted Attributes Policy

The following LDIF file illustrates applying an encrypted attributes policy AE Policy-test-server to a server test-server:

```
dn: cn=test-server, o=novell
changetype: modify
add: encryptionPolicyDN
encryptionPolicyDN: cn=AE Policy - test-server, o=novell
```

Deleting Encrypted Attributes Policy

The following LDIF file illustrates deleting an encrypted attributes policy:

```
dn: cn=AE Policy - test-server, o=novell
changetype: delete
```

NOTE: For more information on managing encrypted attributes through LDAP, see "Using LDAP Tools on Linux" on page 349 and "OpenText Import Conversion Export Utility" on page 157.

Copying the Encrypted Attributes Policies

You can copy the encrypted attributes policies to have identical configurations on many servers. The policies are stored as objects in OpenText eDirectory.

Refer to "Copying Objects" on page 98 for step-by-step information on copying a Policy object using OpenText Identity Console.

Partition Operations

When you merge two partitions, the policies of the parent are retained for the resultant partition. When you split a partition, the child partition inherits the policy of the parent partition.

Recommendation: OpenText eDirectory stores several attributes for its own operations which should not be marked for encryption. If these attributes are marked for encryption, some of the OpenText eDirectory functionality will possibly be broken or it will not perform as expected.

The attributes that should not marked for encryption are:

- federationBoundaryType
- Volume
- ACL
- federationBoundary
- member
- federationControl
- federationSearchPath
- encryptionPolicyDN
- indexDefinition
- dgIdentity

- dgAllowUnknown
- agTimeout
- Host Server
- hostResourcePath
- ndsPredicateState
- ndsStatusExternalReference
- ndsStausLimber
- ndsStatusSchema

Though the list is not exhaustive, similar kind of attributes should not be marked for encryption.

Accessing the Encrypted Attributes

When you encrypt the attributes, you also protect the access to the encrypted attributes. This is because OpenText eDirectory can restrict the access to the encrypted attributes over secure channel such as LDAP secure channel or NCP secure channel. However, only OpenText internal customers can set up and use a secure NCP connection because the DClient application, with which a secure NCP connection is created, is not available for public use.

You can also back up the encrypted attributes by using the Backup (ndsbackup) utility.

By default, the encrypted attributes can be accessed only through a secure channel.

However, if you want the clients to be able to access the encrypted attributes over clear text, then disable the Always Require Secure Channel option. For more information, refer to "Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels" on page 300.

NOTE: When a non-secure search is performed on an object which has an encrypted attribute, the non-encrypted attributes will not be returned unless they are explicitly specified.

Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels

You can enable or disable the access to encrypted attributes over clear text channels by enabling or disabling Always Require Secure Channel option (that is, the attrEncryptionRequireSecure attribute) using either OpenText Identity Console or LDAP.

This section contains the following information:

- "Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels Using OpenText Identity Console" on page 301
- "Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels Using LDAP" on page 301

Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels Using OpenText Identity Console

To enable or disable the access to encrypted attributes over clear text channels using OpenText Identity Console, enable or disable Always Require Secure Channel in the Encrypted Attributes Policies Management page while

- Creating and defining encrypted attributes policies.
- Editing encrypted attributes policies.

Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels Using LDAP

To enable or disable access to encrypted attributes over clear text channels using LDAP, add the following attribute to the encrypted attributes policy:

`attrEncryptionRequiresSecure`

Setting this attribute to 0 makes a secure channel not always necessary, that is, you can access the encrypted attributes over a clear text channel. Setting it to 1 makes a secure channel always necessary, that is, you can access the encrypted attributes over a secure channel only.

Refer to Step 3 on page 298 for more information.

Viewing the Encrypted Attributes

Viewing the attributes that are encrypted depends on whether you have enabled or disabled the Always Require Secure Channel option. This means whether you have specified that the encrypted attributes need a secure channel to access them or not.

- “Viewing Encrypted Attributes Using OpenText Identity Console” on page 301
- “Viewing Encrypted Attributes Using DSBrowse” on page 301
- “SNMP Traps” on page 302

Viewing Encrypted Attributes Using OpenText Identity Console

If Always Require Secure Channel is enabled, you cannot view the encrypted attributes. You get the error -6089, indicating that you need a secure channel to access the encrypted attributes.

If Always Require Secure Channel is disabled, you can see the encrypted attributes values in OpenText Identity Console.

For more information, refer to “Browsing Objects in Your Tree” on page 237.

Viewing Encrypted Attributes Using DSBrowse

If you have enabled the Always Require Secure Channel option, that is, if a secure channel is always required to access the encrypted attributes, you cannot view those attributes of the entry that are marked for encryption. However, you can view the other attributes of the entry that are not encrypted.

SNMP Traps

NDS® Value Events are blocked if you have specified that you always need a secure channel to access the encrypted attributes. Traps that are related to value events have value data as NULL and the result will be set to -6089, which indicates that you need a secure channel to get the encrypted attribute value. The following traps have the value data as NULL:

- ndsAddValue
- ndsDeleteValue
- ndsDeleteAttribute

Encrypting and Decrypting Backup Data

While backing up data on a server that has attributes marked for encryption, you are prompted to provide a password to encrypt or decrypt backup data. The `-E` option in the Backup utility facilitates this. For more information, refer to the `ndsbackup` man page.

For more information on backing up your data, refer to Chapter 15, “Backing Up and Restoring OpenText eDirectory,” on page 417.

Cloning the DIB Fileset Containing Encrypted Attributes

While cloning, if the OpenText eDirectory database contains encrypted attributes in it, then the cloned DIB fileset will also have these attribute values encrypted. You need to set a password to secure the key used by OpenText eDirectory to encrypt the values in the cloned DIB fileset. When you place the cloned DIB fileset on another server, you will be asked to provide this password.

For more information, refer to “Clone DIB Set Use Cases” on page 243.

Adding OpenText eDirectory Servers to Replica Rings

You can add OpenText eDirectory servers to replica rings irrespective of whether the attributes are marked for encryption on one or all the servers hosting the replica or whether Always Require Secure Channel is enabled or disabled.

For more information on adding OpenText eDirectory server to the replica ring, refer to “Adding a Replica” on page 147.

Backward Compatibility

You need to change all OpenText eDirectory utilities like Identity Console, SNMP, DirXML® and NSureAudit to secure NCP™ to access encrypted attributes. Otherwise, you need to specify that a secure channel is not necessary to access the encrypted attributes. Refer to “Enabling and Disabling Access to Encrypted Attributes Over Clear Text Channels” on page 300 for more information.

Migrating to Encrypted Attributes

When you upgrade OpenText eDirectory, you can encrypt the existing attributes by creating and defining encrypted attributes policies. For more information, refer to “Managing Encrypted Attributes Policies” on page 295.

Replicating the Encrypted Attributes

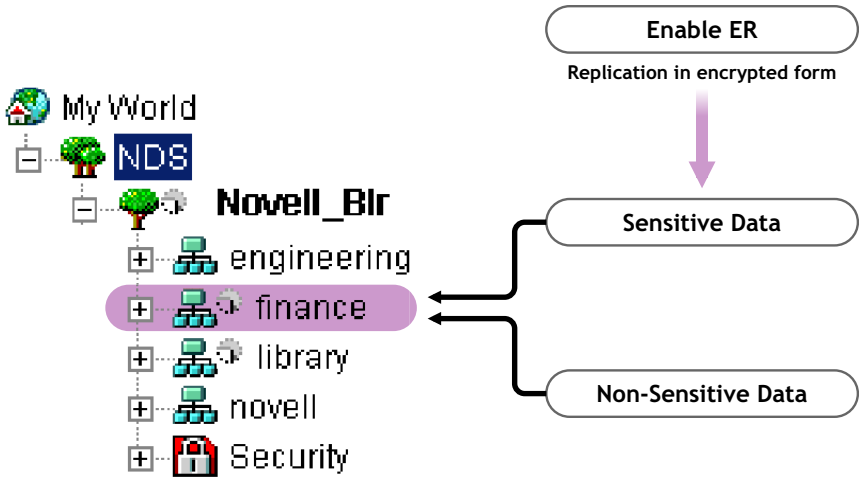
By default, encrypted replication is not enabled even if the server has the encrypted attributes. You need to enable encrypted replication for replicating the encrypted attributes securely. For configuring encrypted replication, refer to “Encrypted Replication” on page 303.

Encrypted Replication

OpenText eDirectory lets you encrypt data that is transmitted between OpenText eDirectory servers. This offers a high level of security during replication as the data does not flow in clear text.

NOTE: We have deprecated the support for Encrypted Replication in OpenText eDirectory 9.2.7 release.

Figure 11-3 *Encrypted Replication*



In Figure 11-3, “finance” and “library” are the partitions in the tree. “finance” might contain sensitive data that requires encryption while replicating. You can enable the partition “finance” for encrypted replication. Partitions like “library” that might not contain sensitive data need not be enabled for encrypted replication.

IMPORTANT: When you enable encrypted replication for a partition, the replication process might slow down. You can enable or disable encrypted replication using OpenText Identity Console.

This section provides the following information:

- “Need for Encrypted Replication” on page 304
- “Enabling Encrypted Replication” on page 304

- “Adding a New Replica to a Replica Ring” on page 308
- “Synchronization and Encrypted Replication” on page 309
- “Viewing the Encrypted Replication Status” on page 309

Need for Encrypted Replication

Prior to OpenText eDirectory 8.8, data was transmitted through the wire during replication in clear text. There was a need to protect confidential data over the wire by encrypting it, especially if the replicas were separated geographically and connected through the Internet.

This feature can be used in the following scenarios:

- If the directory servers are spread across geographical locations through WAN and the Internet and there is a need to encrypt sensitive data on wire.
- If you want only some partitions of your tree to be protected, you can selectively indicate the partitions holding the sensitive data to be encrypted for replication.
- If you require encrypted replication between specific replicas of a partition that contain sensitive data.
- If you feel the network in your setup is hostile, you might want to protect sensitive data during replication.

Enabling Encrypted Replication

To enable encrypted replication, you need to configure a partition for encrypted replication. Configuration settings are stored in the partition Root object.

You can choose to enable encrypted replication at a partition level or replica level.

The configurations at the partition level are overridden by the configurations at the replica level. This means, if encrypted replication is

- Enabled at partition level and disabled for specific replicas, then the replication between the specific replicas happens in clear text.
- Disabled at partition level and enabled for specific replicas, then the replication between the specific replicas happens in encrypted form.

Table 11-1 *Overriding Encrypted Replication Configuration at the Partition Level*

Partition Level	Replica Level	Replication
Enabled	Disabled	Unencrypted
Disabled	Enabled	Encrypted

This section contains the following procedures:

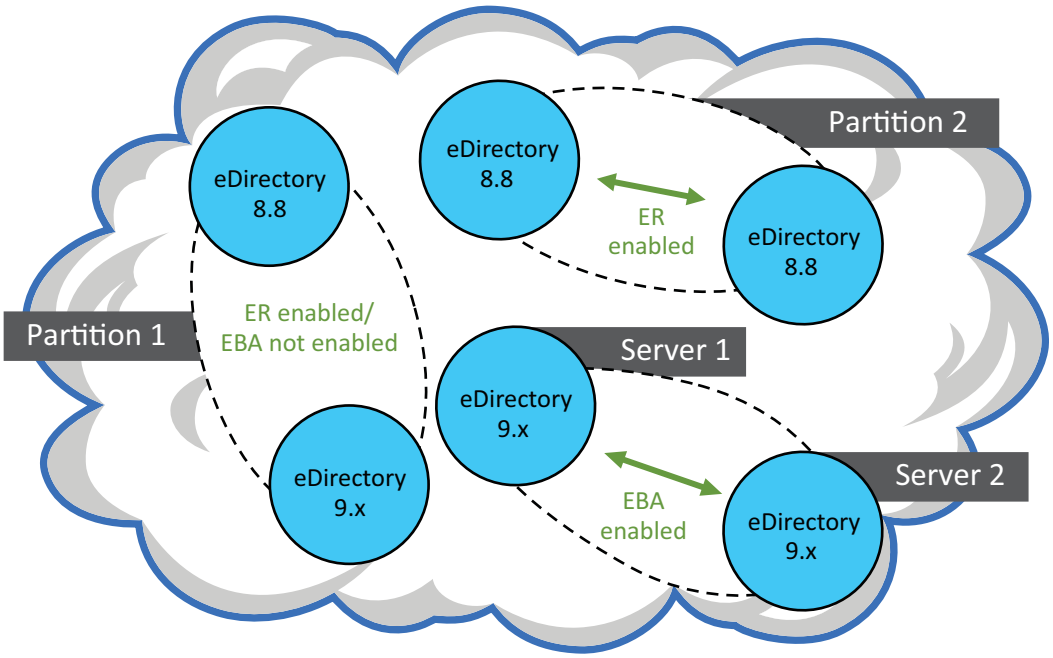
- “Enabling Encrypted Replication at the Partition Level” on page 305
- “Enabling Encrypted Replication at the Replica Level” on page 307

Enabling Encrypted Replication at the Partition Level

When you enable encrypted replication at a partition level, replication between all the replicas hosting the partition is encrypted. For example, consider partition P1 has replicas R1, R2, R3, and R4. You can encrypt the replication between all the replicas, and all replications, inbound or outbound, are encrypted for these replicas.

To enable a partition for encrypted replication, all the servers hosting the partition must be OpenText eDirectory 8.8 or later servers.

Figure 11-4 *Encrypted Replication*



The configurations for encrypted replication at the partition level are overridden if you have encrypted replication configurations at replica level. Refer to Table 11-1 on page 304.

Backward compatibility depends on whether the encrypted replication is enabled or disabled at the partition level. Refer to “Adding a New Replica to a Replica Ring” on page 308 for more information.

You can enable encrypted replication at the partition level using OpenText Identity Console or LDAP, as explained in the following sections:

- “Enabling Encrypted Replication at the Partition Level using OpenText Identity Console” on page 305
- “Enabling Encrypted Replication at the Partition Level Using LDAP” on page 306
- “Partition Operations” on page 308

Enabling Encrypted Replication at the Partition Level using OpenText Identity Console

1 On the OpenText Identity Console home page, click **Encrypted Replication** tile.

2 On the **Encrypted Replication** page > click **Search Partition** .

3 Select the tree or container.

4 Select the check box **Enable Encrypted replication** > Click **Finish**.

NOTE: To disable encrypted replication at the partition level, unselect the check box **Enable Encrypted replication**.

---

5 The **Encrypted Replication successfully enabled** message appears.

Click **OK**.

In the Encrypted Replication page, when you enable encrypted replication for the whole partition, you can disable encrypted replication for specific replicas. The replicas that you disable for encrypted replication will not receive or send data in encrypted form. You can also disable encryption for the entire partition by deselecting **Enable Encrypted replication**.

Enabling Encrypted Replication at the Partition Level Using LDAP

IMPORTANT: We strongly recommend you to use OpenText Identity Console for enabling encrypted replication.

To encrypt replication, you need to use the attribute dsEncryptedReplicationConfig. The syntax is:

```
enable/disable flag#destination replica number#source replica number
```

Replace with either of these flags:

- 0: Encrypted replication is disabled
- 1: Encrypted replication is enabled

Source replica number and destination replica number represents source and destination replica numbers of a partition. These numbers can be specified in any order because if the replication from A to B is encrypted, then replication from B to A is also encrypted.

NOTE: If the source and destination replica number at the partition level is 0 and if the flag is set to 1, all the replicas are considered to be enabled for encrypted replication.

To enable encrypted replication at the partition level, the value of the dsEncryptedReplicationConfig attribute should be set to 1#0#0.

Following is a sample LDIF file for enabling encrypted replication at the partition level:

```
dn: o=ou
changetype:modify
replace: dsEncryptedReplicationConfig
dsEncryptedReplicationConfig:1#0#0
```

These configurations at the partition level are overridden by the configurations at the replica level. Refer to “Enabling Encrypted Replication at the Replica Level using LDAP” on page 307 for more information.

Enabling Encrypted Replication at the Replica Level

When you enable encrypted replication at the replica level, replication between specific replicas is encrypted. Both outbound and inbound replication between the replicas are encrypted.

For example, consider partition P1 has replicas R1, R2, R3, and R4. You can encrypt the replication between replicas R1 and R2 or between R2 and R4.

If you have enabled encrypted replication for one replica, it means that:

- the inbound synchronization from a server to this replica
- outbound synchronization from this replica to any other server is encrypted.

The replicas you have enabled for encrypted replication must be on OpenText eDirectory 8.8 or later servers. The remaining replicas in the replica ring, that are not enabled for encrypted replication, can be on servers with earlier versions of OpenText eDirectory.

To disable encrypted replication at the replica level, you need to disable **Encrypt Link** for specific replicas using Encrypted Replication Configuration page in OpenText Identity Console.

You can enable encrypted replication at the replica level using LDAP as described in the following section:

- "Enabling Encrypted Replication at the Replica Level using LDAP" on page 307

Enabling Encrypted Replication at the Replica Level using LDAP

IMPORTANT: We strongly recommend you to use OpenText Identity Console for enabling encrypted replication.

To encrypt replication, you need to use the attribute `dsEncryptedReplicationConfig`. The syntax is:

```
enable/disable flag#destination replica number#source replica number
```

For more information on the syntax, refer to "Enabling Encrypted Replication at the Partition Level Using LDAP" on page 306.

When you specify the replicaNumber of the replicas in the above syntax, you enable the encrypted replication between those replicas. consider the following example syntaxes:

- 1#0#1: Encrypted replication is enabled from and to replica number 1; to and from, every other replica in the partition.
- 0#3#1: Encrypted replication is disabled between replica numbers 3 and 1.
- 0#1#1: Encrypted replication is disabled for replica number 1.

The following is a sample LDIF file that disables encrypted replication between replica numbers 1 and 3:

```
dn: o=ou
changetype: modify
replace: dsEncryptedReplicationConfig
dsEncryptedReplicationConfig: 0#3#1
```

Partition Operations

When you split a partition, the encrypted replication configuration in the parent partition is inherited by the child partition. When you merge a partition, the encrypted replication configuration of the parent partition is retained in the resultant partition.

Adding a New Replica to a Replica Ring

Adding new replica to a replica ring is affected by whether encrypted replication is enabled or disabled for the partition at the partition and replica level.

For more information on adding a replica to a replica ring, refer to “Administering Replicas” on page 147.

At each of the above levels, you have different scenarios depending on which version of OpenText eDirectory server you are trying to add to the replica ring, as explained in the following sections:

- “Enabling Encrypted Replication at the Partition Level” on page 308
- “Enabling Encrypted Replication at the Replica Level” on page 308
- “Enabling Encrypted Replication for the Server You Add” on page 309

Enabling Encrypted Replication at the Partition Level

The scenarios vary depending on the version of OpenText eDirectory server you are trying to add.

Scenario	Data Encryption
Adding an OpenText eDirectory 9.1 or above server without EBA and with Encrypted Replication disabled	The data flows in clear text.
Adding an OpenText eDirectory 9.1 or above server with Encrypted Replication and without EBA	OpenText eDirectory encrypts data based on the encrypted replication policies.
Adding an OpenText eDirectory 9.1 or above server with EBA	EBA-based encryption will take precedence over encrypted replication.

Enabling Encrypted Replication at the Replica Level

If encrypted replication is enabled between a source replica and specific destination replicas, you can add an OpenText eDirectory 8.8 server or later to the replica ring.

The scenarios vary if encrypted replication is enabled between a source replica and all the other replicas in the replica ring. This is similar to adding replicas to a replica ring with encrypted replication enabled or disabled at the partition level. Refer to “Enabling Encrypted Replication at the Partition Level” on page 308 for more information.

Enabling Encrypted Replication for the Server You Add

If the server you are trying to add is on Linux, you can use the `ndsconfig -E` option to enable encrypted replication on the server. Refer to the ndsconfig man pages for more information.

If the server you are trying to add is on Windows, you can enable the Enable Encrypted Replication option in the installation page.

If the server you are trying to add is on platforms other than Linux, you can enable encrypted replication through OpenText Identity Console or LDAP. Refer to "Enabling Encrypted Replication" on page 304 for more information.

Synchronization and Encrypted Replication

If one replica is enabled for encrypted replication and the configuration changes are not synchronized with the other servers, replication happens in the encrypted form between the replicas. The replicas that are not synced with the configuration changes for encrypted replication continue to sync in clear text.

Even if the encrypted replication configuration has not been synchronized across the replicas, the replication between them will happen in the encrypted form.

Viewing the Encrypted Replication Status

You can view the encrypted replication status through OpenText iMonitor as follows:

1. In OpenText iMonitor, click **Agent Synchronization** in the Assistant frame.
2. Click **Replica Synchronization** for the partition you want to view.

   The replica status information is displayed. The **Encryption Status** field displays whether the link from the replica to which you are currently connected is encrypted or not.

   Basically, there are three scenarios in encryption replication (ER):

   - **ER enabled at partition level:** The replica to which you are connected to shows **Encryption State** is enabled.

     To find out which replica you are connected to, in the replica frame, the one that is not hyper linked is the one you are connected to. If you browse to the other replicas it shows that the **Encryption State** is also marked Enabled.
   - **ER enabled at replica level:** You have enabled ER for all replicas from one particular replica (that is, One to All.) In this case, when you are connected to that replica, its **Encryption State** is marked Enabled.
   - **ER enabled/disabled for a combination of replicas:** ER enabled/disabled for one combination of replicas - You have enabled ER for the whole partition but not for a selected set of servers or vice versa.

     For example, you have enabled ER for partition A that has three replicas 1, 2, and 3 and disabled ER for 1 <--> 3. In this case, if you are connected to replica 1, the **Encryption State** is displayed as:

     Server 1 Enabled

     Server 2

Server 3 Disabled

This means that Server 1 is enabled for encrypted replication to all the servers in the replica ring but 1<-->3 is disabled by the administrator.

Achieving Complete Security While Encrypting Data

The first important basic rule to be followed before encrypting the data is:

No information that would eventually be encrypted should ever be written to the hard disk (or any other media) in the clear.

When you mark existing clear text data for encryption, though the data gets encrypted, the existing clear text data might still be present on some part of hard disk where the DIB resides.

There will be “Left Over” clear text pieces of data in some blocks of database if you try to do following operations:

- Mark existing clear text data for encryption
- Change the encryption scheme of an encrypted attribute

The following sections depict deployment scenarios for encrypted data and steps to ensure that the encrypted data is truly secure:

- “Encrypting Data in an All New Setup” on page 310
- “Encrypting Data in an Existing Setup” on page 311
- “Conclusion” on page 312

Encrypting Data in an All New Setup

In case of a new setup, you would have just installed the operating system and then OpenText eDirectory. It is assured that there is no clear text data present in the hard disk where the DIB resides.

Complete the following steps to ensure that the encrypted data in OpenText eDirectory is truly secure:

1. Plan in advance which attributes you want to encrypt and with what scheme.

   That is, you must decide in advance which attributes you want to encrypt before uploading the data in clear text into the OpenText eDirectory.

   WARNING: Once you have loaded any data into the OpenText eDirectory in the clear, you should not mark an attribute for encryption. Though you can do it, this leads to security problems.

   ---

2. Configure OpenText eDirectory and set the encryption schemes that you want on an attribute.
3. Load your existing data into the new server.

   Bulkloading from an LDIF file or replicating with another server are the two most likely scenarios. Make sure that if you bulk load, you don’t copy the clear text LDIF file onto the same hard disk where the DIB resides.

NOTE: Remember the Rule mentioned: No clear text data can ever be written to the disk.

4 Destroy any existing clear text data

Any disks (or on other media) with the clear text data on it should be securely wiped. This includes things like the clear text LDIF file used to bulk load the server, any other server that was used for replication, or tapes with old backups on them.

Encrypting Data in an Existing Setup

This scenario includes the following:

- "Existing Clear Text Data to Encrypted Data" on page 311
- "Changing the Scheme of the Encrypted Data" on page 312

Existing Clear Text Data to Encrypted Data

You can mark clear text data for encryption and ensure that the data is secure through the following methods:

- "Through Replication" on page 311
- "Through Backup and Restore" on page 312

Through Replication

1 Setup encryption on a new server as follows:

1a Plan in advance which attributes you want to encrypt and with what scheme.

That is, you must decide in advance which attributes you want to encrypt before uploading the data in clear text into the OpenText eDirectory.

WARNING: Once you have loaded any data into the OpenText eDirectory in the clear, you should not mark an attribute for encryption. Though you can do it, this leads to security problems.

1b Start with a clear install (probably including the OS) on a freshly formatted and partitioned disk.

This is to ensure that there is no clear text data on the disk. This means you cannot just take an existing computer which has clear text data previous and re-install OpenText eDirectory. You must have thoroughly erased all traces of data from the disk. Run some kind of secure erase software, use a magnetic bulk eraser on the disk, or perform something equally destructive to the data before installing OpenText eDirectory.

1c Configure OpenText eDirectory and set the encryption schemes that you want on an attribute.

2 Move this server into a replica ring where you have the existing data that you want to encrypt, let the replication happen then take the old server offline.

3 Destroy any existing clear text data

Any disks (or on other media) with the clear text data on it should be securely wiped. This includes things like the clear text LDIF file used to bulk load the server, any other server that was used for replication, or tapes with old backups on them.

Through Backup and Restore

1 Setup encrypting on a new server as follows:

   1a Plan in advance which attributes you want to encrypt and with what scheme.

   That is, you must decide in advance which attributes you want to encrypt before uploading the data in clear text into the OpenText eDirectory.

   WARNING: Once you have loaded any data into the OpenText eDirectory in the clear, you should not mark an attribute for encryption. Though you can do it, this leads to security problems listed in Note A.

   1b Start with a clear install (probably including the operating system) on a freshly formatted and partitioned disk.

   This is to ensure that there is no clear text data on the disk. This means you cannot just take an existing computer which has clear text data previous and re-install OpenText eDirectory. You must have thoroughly erased all traces of data from the disk. Run some kind of secure erase software, use a magnetic bulk eraser on the disk, or perform something equally destructive to the data before installing OpenText eDirectory.

   1c Configure OpenText eDirectory and set the encryption schemes that you want on an attribute.

2 Restore the backed up DIB (that contains the existing clear text data) on the new server. You can backup the DIB using Clone DIB Set or Hot Backup.

3 Destroy any existing clear text data

   Any disks (or on other media) with the clear text data on it should be securely wiped. This includes things like the clear text LDIF file used to bulk load the server, any other server that was used for replication, or tapes with old backups on them.

Changing the Scheme of the Encrypted Data

The steps require to do this using backup/restore are mentioned below:

1 Change the encryption algorithms for an attribute.

2 Take a DIB backup. You can backup the DIB using Clone DIB Set or Hot Backup.

3 Restore the backed up DIB to a new fresh server, and delete the old server.

4 Destroy any existing clear text data on the old server. This avoids bits and pieces of data with the old scheme still on the hard disk.

   Any disks (or on other media) with the clear text data on it should be securely wiped.This includes things like the clear text LDIF file used to bulk load the server, any other server that were used for replication or tapes with old backups on them.

Conclusion

The scenarios listed here are not exhaustive and there might be more scenarios where this problem occurs. As long as you follow the rule, *No information that would eventually be encrypted should ever be written to the hard disk (or any other media) in the clear*, the encrypted data will be truly secure.

12 Repairing the OpenText eDirectory Database

The DSRepair utility lets you maintain and repair the database of an OpenText eDirectory tree. This utility performs the following operations:

- Corrects OpenText eDirectory problems such as bad records, schema mismatches, bad server addresses, and external references.
- Makes advanced changes to the OpenText eDirectory schema.
- Checks the structure of the database automatically without closing the database and without user intervention.
- Checks the database operational indexes.
- Reclaims free space by discarding empty records.
- Repairs the local database.
- Repairs replicas, replica rings, and Server objects.
- Analyzes each server in each local partition for synchronization errors.
- Locates and synchronizes objects in the local database.

Some OpenText eDirectory database problems are not fatal, and OpenText eDirectory will continue to operate. But if the database becomes corrupted, you will get a message on the console that the server could not open the local database. In this case, run Repair or contact OpenText Support.

OpenText does not recommend running repair operations unless you run into problems with OpenText eDirectory, or are told to do so by OpenText Support. However, you are encouraged to use the diagnostic features available in Repair and in other OpenText utilities such as OpenText iMonitor. For more information, see Chapter 8, “Monitoring OpenText,” on page 219.

The list of Repair Wizards:

Wizard	Description
Basic Repair Wizard	Lets you perform an unattended full repair, local database repair, or single object repair. You can also check external references and delete unknown leaf objects.
Log File Wizard	Lets you open the repair log file and set log file options.
Repair via OpenText iMonitor	Lets you open OpenText iMonitor and use the repair options available in that program.
Replica Repair Wizard	Lets you repair all or selected replicas, repair time stamps and declare a new epoch, designate the current server as the new master replica, and destroy the selected replica, if necessary.

Wizard	Description
Replica Ring Repair Wizard	Lets you repair all or selected replica rings, send all objects to every server in the ring, receive all objects from the master to the selected replica, and remove the current server from the replica ring, if necessary.
Schema Maintenance Wizard	Lets you request schema from the tree, reset the local schema, declare a new schema epoch, perform optional schema enhancements, import remote schema, declare a new schema epoch, and perform a schema update.
Server Repair Wizard	Lets you repair all network addresses, or repair only a server's network addresses.
Sync Repair Wizard	Lets you synchronize the selected replica on the current server, report the synchronization status on the current server, report the synchronization status on all servers, perform a time synchronization, and schedule an immediate synchronization.

The wizards help you with the following operations:

- "Performing Basic Repair Operations" on page 314
- "Viewing and Configuring the Repair Log File" on page 317
- "Performing a Repair in OpenText iMonitor" on page 318
- "Repairing Replicas" on page 318
- "Repairing Replica Rings" on page 320
- "Maintaining the Schema" on page 322
- "Repairing Server Network Addresses" on page 325
- "Performing Synchronization Operations" on page 326
- "DSRepair Options" on page 328
- "Using the Client to Repair a Database" on page 333
- "Graphical DS Repair Utility" on page 335

Performing Basic Repair Operations

The Basic Repair Wizard lets you perform an unattended full repair, local database repair, or single object repair. You can also check external references and delete unknown leaf objects.

- "Performing an Unattended Full Repair" on page 315
- "Performing a Local Database Repair" on page 316
- "Checking External References" on page 316
- "Repairing a Single Object" on page 316
- "Deleting Unknown Leaf Objects" on page 317

Performing an Unattended Full Repair

An unattended full repair checks for and repairs most critical OpenText eDirectory errors in the OpenText eDirectory database files of a given server. It performs eight primary operations each time it is run, none of which require any intervention by the administrator. During some of these operations, the local database is locked. An unattended full repair builds a temporary set of local database files and runs the repair operation against those files. That way, if a serious problem develops, the original files are still intact.

Troubleshooting specific issues and resolving them is far superior to running an unattended repair. Running the Unattended Full Repair might require twice the amount of disk space currently used by the database files. See "Performing a Local Database Repair" on page 316 for more information.

Rebuilding the operational indexes used by OpenText eDirectory is possible only when the local database is locked.

The following table lists the operations performed during an unattended full repair:

Operation	Database Locked?	Description
Database Structure and Index Checked	Yes	Reviews the structure and format of database records and indexes. This ensures that no structural corruption has been introduced into the OpenText eDirectory environment at the database level.
Rebuild the Entire Database	Yes	Resolves errors found during structure and index checks. It restores proper data structures and re-creates the OpenText eDirectory database and index files.
Perform Tree Structure Check	Yes	Examines the links between database records to make sure that each child record has a valid parent. This helps ensure database consistency. Invalid records are marked so that they can be restored from another partition replica during the OpenText eDirectory replica synchronization process.
Repair All Local Replicas	Yes	Resolves OpenText eDirectory database inconsistencies by checking each object and attribute against schema definitions. It also checks the format of all internal data structures. This operation can also resolve inconsistencies found during the tree structure check by removing invalid records from the database. As a result, all child records linked through the invalid record are marked as orphans. These orphan records are not lost, but this process could potentially generate a large number of errors while the database is being rebuilt. This is normal, and the orphan objects will be automatically reorganized over the course of replica synchronization.
Repair Network Addresses	No	Checks server network addresses stored in OpenText eDirectory against the values maintained in local SAP, SLP, or DNS tables to make sure that OpenText eDirectory still has accurate information. If a discrepancy is found, OpenText eDirectory is updated with the correct information.

Operation	Database Locked?	Description
Validate Stream Syntax Files	Yes	Stream Syntax Files, such as login scripts, are stored in a special area of the OpenText eDirectory database. This operation checks to make sure that each stream syntax file is associated with a valid OpenText eDirectory object. If not, the stream syntax file is deleted and the attribute referencing it is purged.

Performing a Local Database Repair

Use this repair operation to resolve inconsistencies in the local database so that it can be opened and accessed by OpenText eDirectory.

A local database repair can be performed on a temporary set of files if you specifically request it. Otherwise, the repair operation will take place on the live database.

Performing the repair operation on a temporary set of database files requires closing the database during this part of the operation. If you choose to work on a temporary set of files, you will be prompted to commit the repair modifications before they are made permanent. Otherwise, changes take place immediately.

Following a repair operation, you can view a log of the repair operations to determine if further operations are required to complete the repair. For more information, see “Viewing and Configuring the Repair Log File” on page 317.

Perform a local database repair by using the command: `ndsrepair -J <hexa decimal value of server specific ID>`

Checking External References

This repair operation checks each external reference object to determine if a replica containing the object can be located. If all the servers containing a replica of the partition that the object is in are inaccessible, the object will not be found. If the object cannot be found, a warning is posted.

This operation also provides obituary information.

Check the external references by using the command: `ndsrepair -C`

Repairing a Single Object

This repair operation will try to resolve any inconsistencies in an OpenText eDirectory object which might be preventing OpenText eDirectory from accessing such data. This operation works only on user-created partitions and on the external reference partition.

This operation is performed on the live database files. If the corruption is at the physical level, you might need to perform a Physical and Structure check before the Single Object Repair is run.

Make sure you always have a current backup copy of the OpenText eDirectory database.

Repair single object by using the command: `ndsrepair -J`

Deleting Unknown Leaf Objects

Repair changes inconsistent objects to Unknown objects when they do not have mandatory properties or are invalid in other respects (their properties don't meet minimum requirements for an object type). Unknown objects are real objects and OpenText eDirectory knows about them. They are unknown because their object class cannot be fully validated. Unknown objects, represented by question mark icons, can be deleted but cannot easily be changed back to their original object type.

This repair operation deletes all objects in the local OpenText eDirectory database that have the Unknown object class and maintain no subordinate objects. The deletion is later synchronized to other replicas in the OpenText eDirectory tree.

IMPORTANT: This operation should not be run unless you understand the consequences or have been advised by OpenText Support to run it.

---

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 14.

Viewing and Configuring the Repair Log File

The Repair log file contains detailed information about local partitions and servers. This information helps you diagnose damage to the database. The Log File Wizard lets you open the repair log file and set log file options.

This sections contains information on the following operations:

- "Opening the Log File" on page 317
- "Setting Log File Options" on page 318

Opening the Log File

Use this operation to view your repair log file. The default name of the file is `dsrepair.log`. The results of the operations performed by your repairs are written to it.

You can turn the log file operation off or on, change the name, and delete or reset the log file. See "Setting Log File Options" on page 318 for more information.

1. Login to `eMBox client` > type `list` > press the Enter key.
2. Type `getloginfo` > press the Enter key.

   You can view the repair log file.

Setting Log File Options

Use this operation to manage the repair log file. You can turn the log file on or off, delete the log file, append the log file, or change the filename.

1. Login to `eMBox client` > type `list` > press the Enter key.
2. Type `setloginfo -f filename.log` > press the Enter key.

Performing a Repair in OpenText iMonitor

You can access Repair features by using the **iMonitor**. The Repair page in iMonitor lets you view problems and back up or clean up your OpenText eDirectory database.

In OpenText iMonitor, DSRepair is a server-centric feature. In other words, this feature is available only on the local server where OpenText iMonitor is running. If you need to access this feature on another server, you must switch to the OpenText iMonitor running on that server.

You must be the equivalent of the Administrator of the server or a console operator on the server where you are trying to access the DS Repair page. For this reason, you must first log in so your credentials can be verified before you can access information on this page.

For more information on using the repair features available in iMonitor, see “Viewing DSRepair Information” on page 236.

Repairing Replicas

Repairing a replica consists of checking each object in the replica for consistency with the schema, and checking each attribute of the object for consistency with the schema and the data according to the syntax of the attribute. Other internal data structures associated with the replica are also checked.

Use the Replica Repair Wizard to perform the following operations:

- “Repairing All Replicas” on page 318
- “Repairing Selected Replicas” on page 319
- “Repairing Time Stamps” on page 319
- “Designating This Server As the New Master Replica” on page 320
- “Destroying the Selected Replica” on page 320

Repairing All Replicas

This operation repairs all of the replicas displayed in the replica table.

If you have not performed a Local Database Repair operation on the local OpenText eDirectory database within the last 30 minutes, you should do so before performing this operation. See “Performing a Local Database Repair” on page 316 for more information.

To repair all replicas, run the command: `ndsrepair -R`

Repairing Selected Replicas

This operation repairs only the selected replica listed in the replica view.

If you have not performed a Local Database Repair operation on the local OpenText eDirectory database within the last 30 minutes, you should do so before performing this operation. For more information see "Performing a Local Database Repair" on page 316.

For repairing selected replicas, see the section: DSRepair eMTool Options.

Repairing Time Stamps

NOTE: Before using this operation, use the Sync Repair Wizard to make sure that all servers in the replica ring are communicating properly. See "Performing Synchronization Operations" on page 326 for more information.

This operation provides a new point of reference to the master replica so that all updates to replicas of the selected partition are current.

This operation is always performed on the master replica of a partition. The master replica does not need to be the local replica on this server.

Time stamps are placed on objects when they are created or modified, and they must be unique. All time stamps in a master replica are examined. If any time stamps are postdated to the current network time, they are replaced with a new time stamp. If the time stamp is current, a new time stamp is not issued. After all time stamps are consistent in time, a new epoch is declared.

Use this operation if you notice a discrepancy between objects in a replica or in an object's properties. For example, if you update your login script but your old login script still appears when logging in, you should check to ensure that replicas are synchronizing properly. If the differences between the time stamps in the future and the current time is not more than minutes, OpenText eDirectory will eventually correct the condition by itself. Declaring a new epoch is a very expensive operation, and should not be used regularly.

OpenText eDirectory is a loosely consistent database, so you should allow for five to ten minutes before checking replica synchronization. This operation results in the following conditions:

- A new epoch is declared on the master replica, possibly affecting all objects in the replica.
- All time stamps are examined and repaired as required.
- Updates are not accepted from replicas with postdated time stamps (epochs) until the replicas are synchronized.
- A replica receives a copy of all objects in a master replica or any other replica that has received a new epoch.
- The replica becomes the same epoch as the master replica.
- Any modifications from a previous epoch are lost.
- The master replica does not need to reside on the current server, but you must have the Supervisor right to the master replica to perform the repair operation.
- The other replicas are put in a new state.

To repair time stamps and declare a new epoch, see the section: DSRepair eMTool Options.

Designating This Server As the New Master Replica

This operation designates the local replica of the selected partition as the master replica. You can use this operation to designate a new master replica if the original one is lost. A master can be lost if the server that contains the master replica has a hard disk failure and must be replaced.

For more information, see Chapter 6, “Managing Partitions and Replicas,” on page 143.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 5.
4. Specify the administrator name and password.

Destroying the Selected Replica

Use this operation to remove the selected replica from this server. The replica will be deleted or changed to a subordinate reference.

For more information, see Chapter 6, “Managing Partitions and Replicas,” on page 143.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 13.
4. Respond to the warning message and specify the administrator name and password details.

Repairing Replica Rings

Repairing a replica ring consists of checking the replica ring information on each server that contains a replica and validating remote ID information.

Use the Replica Ring Repair Wizard to perform the following operations:

- “Repairing All Replica Rings” on page 321
- “Repairing the Selected Replica Ring” on page 321
- “Sending All Objects to Every Server in the Ring” on page 321
- “Receiving All Objects from the Master to the Selected Replica” on page 322
- “Removing This Server from the Replica Ring” on page 322

Repairing All Replica Rings

This operation repairs the replica ring of all the replicas displayed in the replica view.

If you have not performed a Local Database Repair operation on the local OpenText eDirectory database within the last 30 minutes, you should do so before performing this operation. See "Performing a Local Database Repair" on page 316 for more information.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 8.

Repairing the Selected Replica Ring

This operation repairs the replica ring of the selected replica listed in the replica table.

If you have not performed a Local Database Repair operation on the local OpenText eDirectory database within the last 30 minutes, you should do so before performing this operation. See "Performing a Local Database Repair" on page 316 for more information.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 9.

Sending All Objects to Every Server in the Ring

This operation sends all objects from the selected server in the replica ring to all other servers that contain a replica of the partition.

Use this operation to ensure that the selected partition's replica on the selected server in the replica ring is synchronized with all other servers in the replica ring. This operation cannot be performed on a server that contains only a subordinate reference replica of the partition.

Modifications that have been made to other replicas that have not yet synchronized with the replica on the selected server will be lost. You should verify the synchronization status before performing this operation.

IMPORTANT: This operation can cause heavy network traffic because of the re-creation of the objects in the replica. It is not a diagnostic operation.

---

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 10.
4. Provide the server details.
5. Select 3.
6. Specify administrator name and password.

Receiving All Objects from the Master to the Selected Replica

This operation receives all objects from the master replica to the replica on the selected servers.

Use this operation to ensure that the selected partition's replica on the selected server in the replica ring is synchronized with the master replica. This operation cannot be performed on a server that contains the master replica.

IMPORTANT: This operation can produce a lot of network traffic. By requesting this operation, the current replica will behave as if a new replica is being placed on the server. It will also put the replica in a new state.

---

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 10.
4. Provide the server details.
5. Select 4.
6. Specify administrator name and password.

Removing This Server from the Replica Ring

This operation removes the specified server from the selected replica stored on the current server.

WARNING: Misuse of this operation can cause irrevocable damage to the OpenText eDirectory database. You should not use this operation unless directed to by OpenText Support personnel.

---

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 10.
4. Provide the server details.
5. Select 6.
6. Specify administrator name and password.

Maintaining the Schema

The schema is a system of rules and definitions for object attributes that establishes the content and format of each object and the object's relationships in the database.

The Schema Maintenance Wizard contains several schema operations that might be necessary to bring an OpenText eDirectory server's schema into compliance with the master of [Root]. However, these operations should be used only when necessary. The local and unattended repair operations already verify the schema.

For more information on the OpenText eDirectory schema, see Chapter 5, "Managing the Schema," on page 131.

Use the Schema Maintenance Wizard to perform the following operations:

- "Requesting Schema from the Tree" on page 323
- "Resetting the Local Schema" on page 323
- "Performing Optional Schema Enhancements" on page 323
- "Importing Remote Schema" on page 324
- "Declaring a New Schema Epoch" on page 324

Requesting Schema from the Tree

Use this operation to request the master replica of the root of the tree to synchronize its schema to this server. Any changes to the schema will be propagated to this server from the master replica of the [Root] for the next 24 hours.

IMPORTANT: If all servers request the schema from the master replica, network traffic can increase. Therefore, use this option with caution.

---

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.
3. Choose the option: 1. Requested schema from master server.

Resetting the Local Schema

This operation invokes a schema reset which clears the time stamps on the local schema and requests an inbound schema synchronization.

This operation is unavailable if executed from the master replica of the [Root] partition. This is to ensure that not all servers in the tree reset at once.

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.
3. Choose the option: 2. Reset local schema.

Performing Optional Schema Enhancements

This operation extends and modifies the schema for containment and other schema enhancements.

This operation requires that this server contain a replica of the [Root] partition and that the state of the replica must be On.

Previous versions of OpenText eDirectory cannot synchronize these changes.

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.
3. Choose the option: 3. Optional Schema Enhancements.

Importing Remote Schema

This operation lets you select an OpenText eDirectory tree that contains the schema you want to add to the current tree's schema.

After you select a tree, the server that holds the master replica of the [Root] partition is contacted. The schema from that server is used to extend the schema on the current tree.

In order to merge two trees, you might need to import the schema from one tree to the other more than once.

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.
3. Choose option 4 Import schema from Tree (Global Schema Options).
4. Provide the IP address or tree name.

Declaring a New Schema Epoch

An epoch is an instant in time that is arbitrarily selected as a point of reference. It is synonymous with era or new version. Epochs control the synchronization of replicas. When a new epoch is declared, it begins on the master replica. Other replicas cannot send updates to a replica with a newer epoch, but they receive updates from it until they become fully synchronized with it.

When other replicas of a given partition are synchronized with the updated replica, meaning that each replica's epoch is the same, bidirectional synchronization is allowed again.

When you declare a new schema epoch, the master replica of the [Root] partition is contacted and illegal time stamps are repaired on the schema records. A new epoch for the schema is then declared on that server, but it affects the entire tree.

All other servers receive a new copy of the schema including the repaired time stamps.

If the receiving server contains a schema that was not in the new epoch, objects and attributes that use the old schema are changed to the Unknown object class or attribute.

IMPORTANT: Do not perform this operation unless instructed to do so by OpenText Support.

---

1. Run `Ndsrepair -S -Ad`.
2. Provide administrator name and password.
3. Choose the option: 5. Declare a new epoch.

Repairing Server Network Addresses

The Server Repair Wizard lets you repair all server network addresses in replica rings and Server objects in the local database. You can also repair a selected server's network address in replica rings and Server objects in the local database.

Use the Server Repair Wizard to perform the following operations:

- “Repairing All Network Addresses” on page 325
- “Repairing a Server's Network Addresses” on page 325

Repairing All Network Addresses

This operation checks the network address for every server in the local OpenText eDirectory database. It searches the SAP tables, the SLP directory agent, and DNS local or remote information, depending on the transport protocol available, for each server’s name.

Each address is then compared to the OpenText eDirectory Server object’s Network Address attribute and the address record in each Replica attribute of every partition [Root] object. If the addresses are different, they are updated to be the same.

If the server address cannot be found in the SAP tables, local/remote DNS information, or SLP directory agents, no repair is performed.

1. Run the command `ndsrepair -N -Ad`.
2. From the server list, select the required one.
3. From the server options list, select 1: Repair all network addresses.

Repairing a Server's Network Addresses

This operation checks the network address for the selected server in the local OpenText eDirectory database files. It searches the local SAP tables, the SLP directory agent, or local or remote DNS information, depending on the transport protocols currently bound, for the server’s name. The server’s address is then compared to the OpenText eDirectory Server object’s Network Address attribute and the address record in each Replica attribute of every partition [Root] object. If the addresses are different, they are updated to be the same.

If the server address cannot be found in the SAP tables, SLP, or local/remote DNS information, no repair is performed.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, select replica option.
3. From the replica options list, select 2.

Issues

OpenText SLP is an optional package. The authentication feature is not implemented as a part of the OpenText SLP package.

OpenText eDirectory is now compatible with OpenSLP, and the authentication features of OpenSLP are used.

NOTE: On Linux, OpenText eDirectory does not listen on all the interfaces except the specific IP that is mentioned in the `nds.conf` file. While adding a new IPV6 address, ensure that the `nds.conf` file is modified with the new address for listener to start and the corresponding referrals to be added.

Performing Synchronization Operations

The Sync Repair Wizard lets you synchronize a selected replica on the current server, report the synchronization status on the current server, report the synchronization status on all servers, perform a time synchronization, and schedule an immediate synchronization.

Use the Sync Repair Wizard to perform the following operations:

- "Synchronizing the Selected Replica on This Server" on page 326
- "Reporting the Synchronization Status on This Server" on page 327
- "Reporting the Synchronization Status on All Servers" on page 327
- "Performing a Time Synchronization" on page 327
- "Scheduling an Immediate Synchronization" on page 328

Synchronizing the Selected Replica on This Server

Use this operation to determine the complete synchronization status of every server that has a replica of the selected partition.

This helps you determine the health of a partition. If all of the servers with a replica of the partition are synchronizing properly, the partition is considered healthy. Each server in the replica ring is contacted, then each server performs an immediate synchronization to every other server in the replica ring.

Servers do not synchronize to themselves. Therefore, the status for the current server's own replica is displayed as Host.

1. Run the command ndsrepair -P -Ad.
2. From the partition list, select replica option.
3. From the replica options list, select 10.
4. Provide the server details.
5. Select 2.
6. Specify administrator name and password.

Reporting the Synchronization Status on This Server

This operation reports the replica synchronization status for every partition that has a replica on the current server.

This operation reads the Synchronization Status attribute from the replica [Root] object on each server that holds replicas of the partitions. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization.

It also displays a warning message if synchronization has not completed within 12 hours.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, choose the partition.
3. From the replica options list, select 10: View Replica Ring.
4. From the list of servers, select the required server.
5. From the server options, select 1: Report synchronization status on the selected server.

Reporting the Synchronization Status on All Servers

Use this operation to determine the replica synchronization status for every partition that has a replica on the current server.

This operation reads the Synchronization Status attribute from the replica [Root] object on each server that holds replicas of the partitions. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization.

It also displays a warning message if synchronization has not completed within twelve hours.

1. Run the command `ndsrepair -P -Ad`.
2. From the partition list, choose the partition.
3. From the replica options list, select 6: Report synchronization status of all servers.

Performing a Time Synchronization

This operation contacts every server known to the local OpenText eDirectory database and requests information about each server's OpenText eDirectory and time synchronization status.

The version of OpenText eDirectory running on each server is reported in the **DS version** field.

The **Replica Depth** field reports a -1 if no replicas are stored on a given server. 0 is reported if the server contains a replica of the [Root] partition. A positive integer is reported if a replica exists on a given server and indicates how many objects away from [Root] the closest replica to [Root] is.

All servers in an OpenText eDirectory tree must be synchronized to the same time source. If all servers are not synchronized to the same time, object synchronization across replicas will not be managed correctly when collisions occur.

The Sync Repair Wizard cannot report the time source for each server, but it does reveal the time server type. This information can then be used to determine if time synchronization is configured properly.

IMPORTANT: You should use OpenText iMonitor to monitor for the “Nearly-In-Sync” time synchronization status instead of using DSRepair. See Chapter 8, “Monitoring OpenText,” on page 219 for more information.

To perform time synchronization, run the command `ndsrepair -T`.

For more information, see “Synchronizing Network Time” on page 92.

Scheduling an Immediate Synchronization

This operation schedules a synchronization of all replicas to occur immediately. Use this operation if you want to review synchronization information without having to wait for the synchronization process to run as normally scheduled.

To schedules a synchronization of all replicas to occur immediately:

1. Run the command ndsrepair -P -Ad.
2. From the partition list, select replica option.
3. From the replica options list, select 3: Scheduling an Immediate Synchronization.

DSRepair Options

The DSRepair utilities for each OpenText eDirectory platform contain some advanced features that are hidden from normal use. These advanced features are enabled through switches when loading the DSRepair utility on various platforms.

- “Running DSRepair on the OpenText eDirectory Server” on page 328
- “DSRepair Command Line Options” on page 330
- “Using Advanced DSRepair Switches” on page 332

Running DSRepair on the OpenText eDirectory Server

- “Windows” on page 328
- “Linux” on page 328

Windows

1. Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**.
2. Click **dsrepair.dlm**, then click `Start`.

   To open DSRepair with advanced options, enter `-a` in the **Startup Parameters** field in the OpenText eDirectory Services Console before you start `dsrepair.dlm`.

Linux

To run DSRepair, enter `ndsrepair` at the server console, using the following syntax:

```
ndsrepair {-U |-E |-C |-P [-Ad] |-S [-Ad]|-N |-T |-J <entry_id> [-Ad -AM
<attribute name>]} [-A <yes/no>] [-O <yes/no>][-F filename] [-h
<local_interface:port>] [--config-file <configuration_file_path>]
```

or

```
ndsrepair -R [-l yes|no] [-u yes|no] [-m yes|no] [-i yes|no] [-f yes|no][-
d yes|no] [-t yes|no] [-o yes|no][-r yes|no] [-v yes|no] [-c yes|no] [-F
filename] [-A yes|no] [-O yes|no]
```

IMPORTANT: The advanced switch [-Ad] should be given as last argument. We recommend that the -Ad advanced switch option be enabled only when instructed by a OpenText Support technician. If the config-file is provided as the argument, then it should be given before the advanced switch [-Ad].

Examples

To perform an unattended repair and log events in the `/root/ndsrepair.log` file, or to append events to the log file if it already exists, enter the following command:

```
ndsrepair -U -A no -F /root/ndsrepair.log
```

To display a list of all global schema operations along with the advanced options, enter the following command:

```
ndsrepair -S -Ad
```

To repair the local database by forcing a database lock, enter the following command:

```
ndsrepair -R -l yes
```

To repair a single object when the entry id of the object is known, enter the following command:

```
ndsrepair -J <entry ID in hex>
```

To repair a particular partition or a replica, enter the following command:

```
ndsrepair -P
```

This command returns a list of all the partitions present on the server. You can choose any of the partitions to get the list of operations that can be performed.

To display information about the free space in the database that can be released for your use, enter the following command:

```
ndsrepair -I
```

To repair network addresses, enter the following command:

```
ndsrepair -N
```

NOTE: The input for the `ndsrepair` command can be redirected from an option file. The option file is a text file that can contain replica and partition operation-related options and suboptions that do not require authentication to the server. Each option or suboption is separated by a new line. Make sure that the contents of the file are in the proper sequence. If the contents are not in the proper sequence, the results will be unpredictable.

DSRepair Command Line Options

Option	Description
-U	Unattended Full Repair option. Instructs DSRepair to run and exit without further user assistance. You can view the log file after the repair has completed to determine what actions DSRepair has taken. This option is not a recommended default normal repair. Troubleshooting specific issues and resolving them is far superior to running an unattended repair.
-P	Replica and Partition Operations option. Lists the partitions that have replicas stored in the current server's OpenText eDirectory database files. The Replica options menu provides options to repair replicas, cancel a partition operation, schedule synchronization, and designate the local replica as the master replica.
-S	Global Schema Operations option. Contains several schema operations that might be necessary to bring the server's schema into compliance with the master of the Tree object. However, these operations should be used only when necessary. The local and unattended repair operations already verify the schema.
-C	Check External Reference Object option. Checks each external reference object to determine if a replica containing the object can be located. If all servers that contain a replica of the partition with the object are inaccessible, the object is not found. If the object cannot be found, a warning is posted.
-E	Report Replica Synchronization option. Reports replica synchronization status for every partition that has a replica on the current server. This operation reads the synchronization status attribute from the replica's Tree object on each server that holds replicas of the partitions. It displays the time of the last successful synchronization to all servers and any errors that have occurred since the last synchronization. A warning message is displayed if synchronization has not completed within twelve hours.
-N	Servers Known to This Database option. Lists all servers known to the local OpenText eDirectory database. If your current server contains a replica of the Tree partition, this server displays a list of all serves in the OpenText eDirectory tree. Select one server to cause the server options to be executed.
-J	Repairs a single object on the local server. You need to provide the Entry ID (in hexadecimal format) of the object you want to repair. You can use this option instead of using the Unattended Repair (-U) option to repair one particular object that is corrupted. The Unattended Repair option can take many hours depending on the size of database. This option helps you save time.

Option	Description
-T	Time Synchronization option. Contacts every server known to the local OpenText eDirectory database and requests information about each server's time synchronization status. If this server contains a replica of the Tree partition, then every server in the OpenText eDirectory tree will be polled. The version of OpenText eDirectory that is running on each server is also reported.
-A	Append to the existing log file. The information is added to the existing log file. By default, this option is enabled.
-O	Logs the output in a file. By default, this option is enabled.
-F *filename*	Logs the output in the specified file.
-R	Repair the Local Database option. Repairs the local OpenText eDirectory database. Use the repair operation to resolve inconsistencies in the local database so that it can be opened and accessed by OpenText eDirectory. This option has suboptions that facilitate repair operations on the database. This option has function modifiers which are explained in the table below.
-I	Displays information about the free space in the database that can be released for your use. OpenText eDirectory allows you to retrieve the empty records and reuse the free space by using the Reclaim option of the ndsrepair command.

The function modifiers used with the -R option are described below:

Option	Description
-l	Locks the OpenText eDirectory database during the repair operation.
-u	Uses a temporary OpenText eDirectory database during the repair operation. It prompts the user to save or discard changes and view the log file.
-m	Maintains the original unrepaired database.
-i	Checks the OpenText eDirectory database structure and the index.
-f	Reclaims the free space in the database.
-d	Rebuilds the entire database.
-t	Performs a tree structure check. Set it to Yes to check all the tree structure links for correct connectivity in the database. Set it to No to skip the check. Default =Yes.
-o	Rebuilds the operational schema.
-r	Repairs all the local replicas.
-v	Validates the stream files.
-c	Checks local references.

Using Advanced DSRepair Switches

WARNING: The features described in this section can cause irreversible damage to your OpenText eDirectory tree if they are used improperly. Use these features only if instructed to do so by OpenText Support personnel.

You should make a full backup of OpenText eDirectory on the server before using any of these features in a production environment. See Chapter 15, “Backing Up and Restoring OpenText eDirectory,” on page 417 for more information.

On Linux, enter `ndsrepair -R -Ad -XK2`.

On Windows, enter these options in the **Startup Parameters** field in NDSConsole before you start `dsrepair.dlm`. See “Running DSRepair on the OpenText eDirectory Server” on page 328 for more information.

Switch	Description
`-P`	Marks all OpenText eDirectory objects of type Unknown as referenced. Referenced objects do not participate in the OpenText eDirectory replica synchronization process.
`-WM`	In many cases, the WM: Registered Workstations attribute will become very high when using ZENworks® 2.0. Running DSRepair with -WM will clear these high values.
`-XK2`	Kills all OpenText eDirectory objects in this server's OpenText eDirectory database. This operation is used to destroy a corrupt replica that cannot be removed in any other way.
`-XK3`	Kills all external references in this server's OpenText eDirectory database. This operation is used to destroy all external references in a nonfunctioning replica. If the references are the source of the problem, OpenText eDirectory can then re-create the references in order to get the replica functioning again.
`-RC`	Backs up the DIB. This option is available only on Windows.
`-OT`	Timestamps obituaries while performing a local database repair. All obituaries are timestamped except INHIBIT MOVE.
`-NLD`	Removes IRF from NLS:License Certificate and NLS:Product Container objects.
`-AM`	Moves the attributes that meet the specific criteria to a different container in the FLAIM database. For more information about which OpenText eDirectory attributes qualify moving to a different container, see FLAIM Attribute Containerization.
`-AH`	Does not create the NDO files when the DIB size is lesser than 1 GB and the older NDO files are more than 72 hours old.

Using the Client to Repair a Database

The OpenText eDirectory Management Toolbox (eMBox) Client is a command line Java client that gives you remote access to DSRepair. Because the Client can be run in batch mode, you can use it to do unattended repairs using the OpenText eDirectory DSRepair eMTool.

The `emboxclient.jar` file is installed on your server as part of OpenText eDirectory. You can run it on any machine with a JVM. For more information on the Client, see "Using the Command Line Client" on page 560.

Using the DSRepair eMTool

1. Run the Client in interactive mode by entering the following at the command line:

   ```
   java -cp path_to_the_file/emboxclient.jar -i
   ```

   (If you have already put the `emboxclient.jar` file in your class path, you only need to enter `java -i`.)

   The Client prompt appears:

   ```
   Client>
   ```

2. Log in to the server you want to repair by entering the following:

   ```
   login -s server_name_or_IP_address -p port_number
   -u username.context -w password -n
   ```

   The port number is usually 80 or 8028, unless you have a Web server that is already using the port. The `-n` option opens a nonsecure connection.

   The Client will indicate whether the login is successful.

3. Enter a repair command, using the following syntax:

   ```
   dsrepair.task options
   ```

   For example, `dsrepair.ufr` performs an unattended full repair.

   `dsrepair.rld -a -v` repairs the local database using the Repair All Local Replicas and Check Local References options.

   A space must be between each switch. The order of the switches is not important.

   The Client will indicate whether the repair is successful.

   See "DSRepair eMTool Options" on page 334 for more information on the DSRepair eMTool options.

4. Log out from the Client by entering the following command:

   ```
   logout
   ```

5. Exit the Client by entering the following command:

   ```
   exit
   ```

DSRepair eMTool Options

The following tables lists the DSRepair eMTool options. You can also use the `list -t dsrepair` command in the Client to list the DSRepair options with details. See "Listing eMTools and Their Services" on page 563 for more information.

Option	Description
`rso -o -d`	Single object repair Object ID in hex Object DN
`rts`	Time synchronization
`rss`	Report synchronization status of all partitions
`rld -l -t -d -p -i -f -c -o -a -m -v`	◆ Repair local database ◆ Lock OpenText eDirectory database during entire repair ◆ Use temporary OpenText eDirectory database during repair ◆ Maintain original unrepaired database ◆ Perform database structure check ◆ Perform database structure and index check ◆ Reclaim database free space ◆ Perform tree structure check ◆ Rebuild operational schema ◆ Repair all local replicas ◆ Validate mail directories and stream files ◆ Check local references
`ufr`	Unattended full repair
`rsn -o -d`	Repair selected server's network address Object ID in hex Object DN
`ran`	Repair all network addresses
`rsr -p -d`	Repair selected replica Partition ID Partition DN
`rer`	Repair every replica
`ror -p -d`	Repair selected replica ring Partition ID Partition DN
`rar`	Repair replica ring, all replicas
`ssa -p -d`	Report the replica synchronization status of all servers Partition ID Partition DN
`cer`	Check external references
`rao -p -d -s -d`	Receive all objects for this replica Partition ID Partition DN Server ID Server DN
`sao -p -d -s -d`	Send all objects to every replica in the ring Partition ID Partition DN Server ID Server DN
`dne -p -d`	Repair time stamps and declare a new epoch Partition ID Partition DN
`sri -p -d`	Schedule immediate synchronization Partition ID Partition DN Server ID Server DN

Option	Description
`sks -p -d -s -d`	Synchronize the replica on the selected server Partition ID Partition DN Server ID Server DN
`ske -p -d`	Synchronize the replica on all servers Partition ID Partition DN
`dsr -p -d`	Destroy the selected replica on this server Partition ID Partition DN
`xsr -p -d -s -d`	Remove this server from the replica ring Partition ID Partition DN Server ID Server DN
`dnm -p -d`	Designate this server as the new master replica Partition ID Partition DN
`dul`	Delete unknown leaf objects

Graphical DS Repair Utility

The Graphical DS Repair Utility has been added to OES 11 SP1. This tool is automatically installed during a new OES 11 SP1 installation.

To invoke the user interface, run the `ndsgrepair` command at the server console. Most of the repair operations that can be performed using the console can be performed using the graphical interface. To navigate all of the help topics such as the menu options, press F1 or click **Help** > **Help Contents** in the UI main menu.

If you are upgrading to OES 11 SP1, perform the following steps to manually select novell-ndsgrepair under the OpenText eDirectory pattern:

1. Open YaST, then select **OES Install and Configuration**.
2. Click Details and select **Novell eDirectory Pattern** on left, then scroll down to bottom of the Packages on the right.
3. Select **novell-ndsgrepair**, click **Accept**, then Next, and then **Finish**.

13 Understanding LDAP Services for OpenText eDirectory

The Lightweight Directory Access Protocol (LDAP) is an Internet communications protocol that lets client applications access directory information. It is based on the X.500 Directory Access Protocol (DAP) but is less complex than a traditional client and can be used with any other directory service that follows the X.500 standard.

LDAP is used most often as the simplest directory access protocol.

Lightweight Directory Access Protocol (LDAP) Services for OpenText eDirectory is a server application that lets LDAP clients access information stored in OpenText eDirectory.

LDAP Services includes OpenText eDirectory features that are available through LDAP:

- Provisioning
- Account Management
- Authentication
- Authorization
- Identity Management
- Notification
- Reporting
- Qualification
- Segmentation

You can give different clients different levels of directory access, and you can access the directory over a secure connection. These security mechanisms let you make some types of directory information available to the public, other types available to your organization, and certain types available only to specified groups or individuals.

The directory features available to LDAP clients depend on the functionality built into the LDAP client and the LDAP server. For example, LDAP Services for OpenText eDirectory lets LDAP clients read and write data in the OpenText eDirectory database if the client has the necessary permissions. Some clients have the capability to read and write directory data, while others can only read it.

Some typical client features let clients do one or more of the following:

- Look up information about a specific person, such as an e-mail address or phone number.
- Look up information for all people with a given last name, or a last name that begins with a certain letter.
- Look up information about any OpenText eDirectory object or entry.
- Retrieve a name, e-mail address, business phone number, and home phone number.
- Retrieve a company name and city name.

The following sections provide information about LDAP Services for OpenText eDirectory:

- "Key Terms for LDAP Services" on page 338
- "Understanding How LDAP Works with OpenText eDirectory" on page 341
- "Using LDAP Tools on Linux" on page 349
- "Extensible Match Search Filter" on page 361
- "LDAP Transactions" on page 362

For more information on LDAP, see the following Web sites:

- OpenLDAP (http://www.openldap.org/)
- An LDAP Roadmap & FAQ (http://www.kingsmountain.com/ldapRoadmap.shtml)

Key Terms for LDAP Services

- "Clients and Servers" on page 338
- "Objects" on page 338
- "Referrals" on page 339

Clients and Servers

LDAP Client— An application (for example, Internet Explorer or the OpenText Import Conversion Export utility).

LDAP Server— A server where `nldap.dlm` (for Windows) or `libnldap.so` (for Linux) is running.

Objects

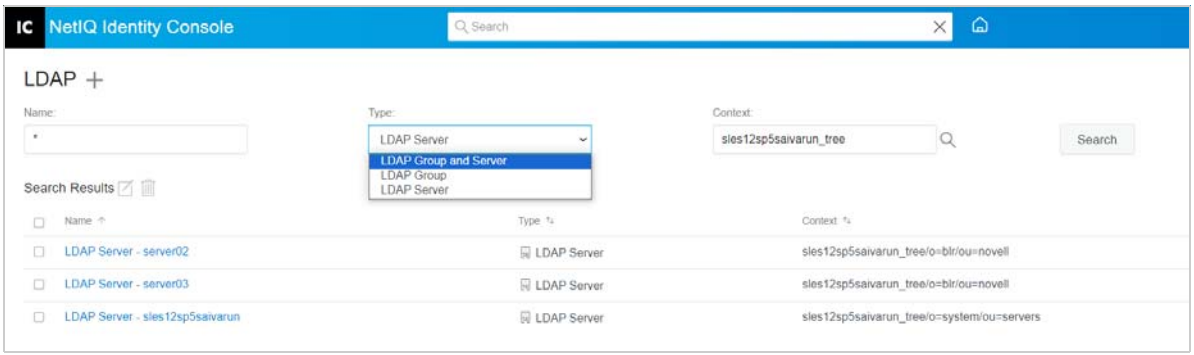
LDAP Group object— Sets up and manages the OpenText LDAP properties on an LDAP server.

This object is created when you install OpenText eDirectory. An LDAP Group object contains configuration information that can be conveniently shared among multiple LDAP servers.

LDAP Server Object— Sets up and manages the way LDAP clients access and use the information on a OpenText LDAP server.

This object is created when you install OpenText eDirectory. An LDAP Server object represents server-specific configuration data.

The following figure illustrates an LDAP Server object in OpenText Identity Console.



Referrals

Referral— A message that the LDAP server sends to the LDAP client telling the client that this server can't provide complete results and that more data might be on another LDAP server.

The referral contains all the information needed to progress the operation.

Scenario: An LDAP client issues a request to an LDAP server but the server can't find the target entry of the operation locally. Using the knowledge references that it has about partitions and other servers, the LDAP server identifies another server that knows more about the entry. The LDAP server sends that information to the client.

The client establishes a new LDAP connection with the identified server and retries the operation.

Referrals have the following advantages:

- The LDAP client keeps control of the operation.

  Because the client always knows what is happening, it can make better decisions and provide feedback to the user. Also, the client can opt not to follow through on a referral, or prompt a user before following it.
- Referrals often use network resources more efficiently than chaining.

  In chaining, a requested search operation with many entries could be transmitted across the network twice. The first transmission would come from the server holding the data to the server doing the chaining. The second transmission would come to the client from the server doing the chaining.

  With a referral, the client gets the data directly from the server that held the data, in one transmission.
- When a client knows where an entry is stored, the client can go directly to the server that has the data.

  Chaining hides details from the client. Not knowing where data came from previously, the client most likely won't go directly to the server holding the data.

Referrals have the following disadvantages:

- The client must be able to recognize referrals and know how to follow them.
- LDAPv2 clients don't recognize referrals, or they use an obsolete, non-standard method for recognizing them.

- Every OpenText eDirectory partition must be serviced by an LDAP server.

  Otherwise, referrals won't be sent for data in that partition.

Superior Referral— A referral to a server that holds data higher in the tree than the server being communicated with. See "Configuring for Superior Referrals" on page 403.

Superior referrals deal with requests concerning objects that are in a higher or contiguous non-OpenText eDirectory partition of a multi-vendor tree.

To enable an OpenText eDirectory server to participate in this type of tree, eDirectory holds the hierarchical data above it in a partition marked as "nonauthoritative." The objects in the non-authoritative area consist only of those entries needed to build the correct DN hierarchy. These entries are analogous to X.500 "Glue" entries.

OpenText eDirectory allows the placement of knowledge information in the form of LDAP referral data within the nonauthoritative area. This information is used to return referrals to the LDAP client.

When an LDAP operation takes place in a nonauthoritative area of the OpenText eDirectory tree, the LDAP server locates the correct reference data and returns a referral to the client.

Chaining— A server-based name-resolution protocol.

An LDAP client issues a request to an LDAP server, but the server can't find the target entry of the operation locally. Using the knowledge references that it has about partitions and other servers in the OpenText eDirectory tree, the LDAP server identifies another LDAP server that knows more about the DN. The first LDAP server then contacts the identified (second) LDAP server.

If necessary, this process continues until the first server contacts a server that holds a replica of the entry. OpenText eDirectory then handles all the details to complete the operation. Unaware of the server-to-server operations, the client assumes that the first server completed the request.

Through chaining, an LDAP server provides the following advantages:

- Hides all name-resolution details from the client
- Automatically takes care of reauthentication
- Acts as a proxy for the client
- Works seamlessly, even when some servers in the OpenText eDirectory tree don't support LDAP Services.

Chaining has the following disadvantages:

- The client might have to wait for some time without any feedback from the server, while the server chains to resolve the name.
- If the operation requires the LDAP server to send many entries across a WAN link, the operation might be very time consuming.
- If several servers are equally capable of progressing the operation, different servers might process two requests to operate on the same entry.

  OpenText eDirectory attempts to sort the servers by the cost associated with contacting them. For load balancing, OpenText eDirectory randomly selects among servers with the lowest cost.

Understanding How LDAP Works with OpenText eDirectory

This section explains the following:

- "Connecting to OpenText eDirectory from LDAP" on page 341
- "Class and Attribute Mappings" on page 344
- "Syntax Differences" on page 347
- "Supported OpenText LDAP Controls and Extensions" on page 349

Connecting to OpenText eDirectory from LDAP

All LDAP clients bind (connect) to OpenText eDirectory as one of the following types of users:

- [Public] User (Anonymous Bind)
- Proxy User (Proxy User Anonymous Bind)
- NDS or OpenText eDirectory User (NDS User Bind)

The type of bind the user authenticates with determines the content that the LDAP client can access. LDAP clients access a directory by building a request and sending it to the directory. When an LDAP client sends a request through LDAP Services for OpenText eDirectory, OpenText eDirectory completes the request for only those attributes that the LDAP client has the appropriate access rights to.

For example, if the LDAP client requests an attribute value (which requires the Read right) and the user is granted only the Compare right to that attribute, the request is rejected.

Standard login restrictions and password restrictions still apply. However, any restrictions are relative to where LDAP is running. Time and address restrictions are honored, but address restrictions are relative to where the OpenText eDirectory login occurred—in this case, the LDAP server.

You can configure a timeout for the write blocked connections from LDAP clients by using the `NDSD_NLDAP_WRITEBLOCK_TIMEOUT` environment variable. This variable defines the time interval for which a connection between a LDAP client and OpenText eDirectory server remains open for the client to read the data from the server. If the connection is blocked and the LDAP client is unable to read the data within the specified time, the connection is considered invalid and subsequently closed. The value must be provided in terms of seconds for this variable as shown in the below example:

```
NDSD_NLDAP_WRITEBLOCK_TIMEOUT=<seconds>
```

```
NDSD_NLDAP_WRITEBLOCK_TIMEOUT=60
```

The default value is 60 seconds.

NOTE: From OpenText eDirectory 9.2.6 and onward, the `NDSD_NLDAP_WRITEBLOCK_TIMEOUT` environment variable is applicable to both read and write blocked connections from LDAP clients. In the previous version, that is OpenText eDirectory 9.2.5, this variable applied to write blocked connections only.

Connecting As a [Public] User

An anonymous bind is a connection that does not contain a user name or password. If an LDAP client without a name and password binds to LDAP Services for OpenText eDirectory and the service is not configured to use a Proxy User, the user is authenticated to OpenText eDirectory as user [Public].

User [Public] is a non-authenticated OpenText eDirectory user. By default, user [Public] is assigned the Browse right to the objects in the OpenText eDirectory tree. The default Browse right for user [Public] allows users to browse OpenText eDirectory objects but blocks user access to the majority of object attributes.

The default [Public] rights are typically too limited for most LDAP clients. Although you can change the [Public] rights, changing them will give these rights to all users. Because of this, we recommend that you use the Proxy User Anonymous Bind. For more information, see "Connecting As a Proxy User" on page 342.

To give user [Public] access to object attributes, you must make user [Public] a trustee of the appropriate container or containers and assign the appropriate object and attribute rights.

Connecting As a Proxy User

A proxy user anonymous bind is an anonymous connection linked to an OpenText eDirectory user name. If an LDAP client binds to LDAP for OpenText eDirectory anonymously, and the protocol is configured to use a Proxy User, the user is authenticated to OpenText eDirectory as the Proxy User. The name is then configured in both LDAP Services for OpenText eDirectory and in OpenText eDirectory.

The anonymous bind traditionally occurs over port 389 in LDAP. However, during the installation you can manually configure different ports.

The key concepts of proxy user anonymous binds are as follows:

- All LDAP client access through anonymous binds is assigned through the Proxy User object.
- Because LDAP clients do not supply passwords during anonymous binds, the Proxy User must have a null password and must not have any password restrictions (such as password change intervals). Do not force the password to expire or allow the Proxy User to change passwords.
- You can limit the locations that the user can log in from by setting address restrictions for the Proxy User object.
- The Proxy User object must be created in OpenText eDirectory and assigned rights to the OpenText eDirectory objects you want to publish. The default user rights provide Read access to a limited set of objects and attributes. Assign the Proxy User Read and Search rights to all objects and attributes in each subtree where access is needed.
- The Proxy User object must be enabled on the General page of the LDAP Group object that configures LDAP Services for OpenText eDirectory. Because of this, there is only one Proxy User object for all servers in an LDAP group. For more information, see "Configuring LDAP Objects" on page 369.
- You can grant a Proxy User object rights to All Properties (default) or Selected Properties.

To give the Proxy User rights to only selected properties:

1. On the OpenText Identity Console home page, click the **Rights Management** tile.
2. Click **Trustee**.
3. Specify, or use the Search Object  to find the name of the object whose trustee list you want to view, then click **OK**.

   The list of the trustees assigned to the selected object is displayed.

   Click Add `Trustee` , to assign a new trustee to the object. Select a Trustee from the list and click  to remove it from the assignment.
4. Select a trustee whose details to be modified as needed, then click **Assigned Rights**.
5. On the **Modify Trustee** screen, assign the rights as needed and click **Done**.
6. On the **Modify Trustee** screen, click **Add Property** , then check the **Show All Properties in Schema** check box.
7. Select an inheritable right for the Proxy User and assign the rights as needed.

   To add additional inheritable rights, repeat Step 6 and Step 7.

   Select a the Property Name from the list and click  to remove it from the assignment.
8. Click **Done**., then click **Apply**.

   The Resource modified successfully! message appears.

To implement proxy user anonymous binds, you must create the Proxy User object in OpenText eDirectory and assign the appropriate rights to that user. Assign the Proxy User Read and Search rights to all objects and attributes in each subtree where access is needed. You also need to enable the Proxy User in LDAP Services for OpenText eDirectory by specifying the same proxy user name.

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **Create Object** .
3. On the Create LDAP Object page > specify the **Name** > select **Type** > search **Context** of an eDirectory User object.
4. Click **Create**, then click **OK**.

Using the ldapconfig Utility on Linux

For example, LDAP Search Referral Usage specifies how the LDAP server processes LDAP referrals.

1. At a system prompt, enter the following command:

   ```
   ldapconfig -s "LDAP:otherReferralUsage=1"
   ```
2. Enter the User FDN (Fully Distinguished eDirectory User Name) and password.

Connecting As an NDS or OpenText eDirectory User

An OpenText eDirectory user bind is a connection that an LDAP client makes using a complete OpenText eDirectory user name and password. The OpenText eDirectory user bind is authenticated in OpenText eDirectory, and the LDAP client is allowed access to any information the OpenText eDirectory user is allowed to access.

The key concepts of OpenText eDirectory user binds are as follows:

- OpenText eDirectory user binds are authenticated to OpenText eDirectory using the user name and password entered at the LDAP client.
- The OpenText eDirectory user name and password used for LDAP client access can also be used for Novell Client access to OpenText eDirectory.
- With non-TLS connections, the OpenText eDirectory password is transmitted in clear text on the path between the LDAP client and LDAP Services for eDirectory.
- If clear text passwords are not enabled, all OpenText eDirectory bind requests that include a user name or password on non-TLS connections are rejected.
- If an OpenText eDirectory user password has expired, OpenText eDirectory bind requests for that user are rejected.

Assigning OpenText eDirectory Rights for LDAP Clients

1. Determine the type of user name the LDAP clients will use to access OpenText eDirectory:
   - [Public] User (Anonymous Bind)
   - Proxy User (Proxy User Anonymous Bind)
   - NDS User (NDS User Bind)

   See "Connecting to OpenText eDirectory from LDAP" on page 341 for more information.
2. When the users use one proxy user, or multiple OpenText eDirectory user names to access LDAP, use the LDAP to create these user names in OpenText eDirectory.
3. Assign the appropriate OpenText eDirectory rights to the user names that LDAP clients will use.

The default rights that most users receive provide limited rights to the user's own object. To provide access to other objects and their attributes, you must change the rights assigned in OpenText eDirectory.

When an LDAP client requests access to an OpenText eDirectory object and attribute, OpenText eDirectory accepts or rejects the request based on the LDAP client's OpenText eDirectory identity. The identity is set at bind time.

Class and Attribute Mappings

A *class* is a type of object in a directory, such as a user, server, or group. An attribute is a directory element that defines additional information about a specific object. For example, a User object attribute might be a user's last name or phone number.

A *schema* is a set of rules that defines the classes and attributes allowed in a directory and the structure of a directory (where the classes can be in relation to one another). Because the schemas of the LDAP directory and the OpenText eDirectory directory are sometimes different, mapping LDAP

classes and attributes to the appropriate OpenText eDirectory objects and attributes might be necessary. These mappings define the name conversion from the LDAP schema to the OpenText eDirectory schema.

LDAP Services for OpenText eDirectory provides default mappings. In many cases, the correspondence between the LDAP classes and attributes and the OpenText eDirectory object types and properties is logical and intuitive. However, depending on your implementation needs, you might want to reconfigure the class and attribute mapping.

In most instances, the LDAP class to OpenText eDirectory object type mapping is a one-to-one relationship. However, the LDAP schema supports alias names such as CN and commonName that refer to the same attribute.

Mapping LDAP Group Attributes

The default LDAP Services for OpenText eDirectory configuration contains a predefined set of class and attribute mappings. These mappings map a subset of LDAP attributes to a subset of OpenText eDirectory attributes. If an attribute is not already mapped in the default configuration, an auto-generated map is assigned to the attribute. Also, if the schema name is a valid LDAP name with no spaces or colons, no mappings are required. You should examine the class and attribute mapping and reconfigure as needed.

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **LDAP Group**.
3. On the Modify LDAP Group page > click **Attribute Map** drop-down.
4. Add, delete, or modify the attributes you want.

   Because there might be alternate names for certain LDAP attributes (such as CN and common name), you might need to map more than one LDAP attribute to a corresponding OpenText eDirectory attribute name. When LDAP Services for OpenText eDirectory returns LDAP attribute information, it returns the value of the first matched attribute it locates in the list.

   If you map multiple LDAP attributes to a single OpenText eDirectory attribute, you should reorder the list to prioritize which attribute should take precedence because the order is significant.
5. Click **Save**.

Class Mapping in LDAP Groups

When an LDAP client requests LDAP class information from the LDAP server, the server returns the corresponding OpenText eDirectory class information. The default LDAP Services for OpenText eDirectory configuration contains a predefined set of class and attribute mappings.

NOTE: OpenText eDirectory does not propagate class mappings in LDAP Group objects across LDAP servers. To use the same class mapping on more than one server, manually add the mapping to all LDAP group objects in your environment.

---

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **LDAP Group**.
3. On the Modify LDAP Group page > click **Class Map** drop-down.

4 Add, delete, or modify the classes you want.

The default LDAP Services for OpenText eDirectory configuration contains a predefined set of class and attribute mappings. These mappings map a subset of LDAP classes and attributes to a subset of OpenText eDirectory classes and attributes. If an attribute or class is not mapped in the default configuration, an auto-generated map is assigned to the attribute or class.

Also, if the schema name is a valid LDAP name with no spaces or colons, no mappings are required. You should examine the class and attribute mapping and reconfigure as needed.

5 Click **Save**.

Mapping LDAP Classes and Attributes

Because the schemas of the LDAP directory and the OpenText eDirectory directory are different, mapping LDAP classes and attributes to the appropriate OpenText eDirectory objects and attributes is necessary. These mappings define the name conversion from the LDAP schema to the OpenText eDirectory schema.

No LDAP schema mappings are required for a schema entry if the name is a valid LDAP schema name. In LDAP, the only characters allowed in a schema name are alphanumeric characters and hyphens (-). No spaces are allowed in an LDAP schema name.

To ensure that searching by object IDs works after a schema extension other than LDAP, such as for .sch files, you must refresh the LDAP server configuration if the schema is extended outside of LDAP.

Many-to-One Mappings

To support LDAP from OpenText eDirectory, LDAP Services uses mappings in the protocol level (instead of the directory service level) to translate between LDAP and OpenText eDirectory attributes and classes. Because of this, two LDAP classes or attributes can be mapped to the same OpenText eDirectory class or attribute.

For example, if you create a Cn through LDAP and then search for CommonName=Value, you will get back a commonName, which might be the same attribute value for Cn.

If you request all attributes, you get the attribute that is first in the mappings list for that class. If you ask for an attribute by name, you will get the correct name.

Many-to-One Class Mappings

LDAP Class Name	eDirectory Class Name
alias aliasObject	Alias
groupOfNames groupOfUniqueNames group	Group
mailGroup rfc822mailgroup	NSCP:mailGroup1

Many-to-One Attribute Mappings

LDAP Attribute Name	eDirectory Attribute Name
c countryName	C
cn commonName	CN
uid userId	uniqueID
description multiLineDescription	Description
l localityname	L
member uniqueMember	Member
o organizationname	O
ou organizationalUnitName	OU
sn surname	Surname
st stateOrProvinceName	S
certificateRevocationList;binary certificateRevocationList	ndspkiCertificateRevocationList
authorityRevocationList;binary authorityRevocationList	authorityRevocationList
deltaRevocationList;binary deltaRevocationList	deltaRevocationList
cACertificate;binary cACertificate	cACertificate
crossCertificatePair;binary crossCertificatePair	crossCertificatePair
userCertificate;binary userCertificate	userCertificate

NOTE: The attributes with `;binary` are security related. They are in the mapping table in case your application needs the name retrieved with `;binary`. If you need it retrieved without `;binary`, you can change the order of the mappings.

Syntax Differences

LDAP and OpenText eDirectory use different syntaxes. Some important differences include the following:

- “Commas” on page 348
- “Typeful Names” on page 348
- “Escape Character” on page 348
- “Multiple Naming Attributes” on page 348

Commas

LDAP uses commas as delimiters rather than periods. For example, a distinguished (or complete) name in OpenText eDirectory looks like this:

CN=JANEB.OU=MKTG.O=EMA

Using LDAP syntax, the same distinguished name would be

CN=JANEB,OU=MKTG,O=EMA

Some additional examples of LDAP distinguished names:

CN=Bill Williams,OU=PR,O=Bella Notte Corp
CN=Susan Jones,OU=Humanities,O=University College London,C=GB

Typeful Names

OpenText eDirectory uses both typeless (.JOHN.MARKETING.ABCCORP) and typeful (CN=JOHN.OU=MARKETING.O=ABCCORP) names. LDAP uses only typeful names with commas as the delimiters (CN=JOHN,OU=MARKETING,O=ABCCORP).

Escape Character

The backslash (\) is used in LDAP distinguished names as an escape character. If you use the plus sign (+) or the comma (,), you can escape them with a single backslash character.

For example:

CN=Pralines\+Cream,OU=Flavors,O=MFG (CN is Pralines+Cream)

CN=DCardinal,O=Lionel\,Turner and Kaye,C=US (O is Lionel, Turner, and Kaye)

See Internet Engineering Task Force RFC 2253 (http://www.ietf.org/rfc/rfc2253.txt?number=2253) for more information.

Multiple Naming Attributes

Objects can be defined with multiple naming attributes in the schema. In both LDAP and OpenText eDirectory, the User object has two: CN and UID. The plus sign (+) separates the naming attributes in the distinguished name. If the attributes are not explicitly labeled, the schema determines which string goes with which attribute (the first would be CN, the second is UID for eDirectory and LDAP). You can reorder them in a distinguished name if you manually label each portion.

For example, the following are two relative distinguished names:

Smith (CN is Smith CN=Smith)

Smith+Lisa (CN is Smith, the UID is Lisa CN=Smith UID=Lisa)

Both relative distinguished names (Smith and Smith+Lisa) can exist in the same context because they must be referenced by two completely different relative distinguished names.

Supported OpenText LDAP Controls and Extensions

The LDAP 3 protocol allows LDAP clients and LDAP servers to use controls and extensions for extending an LDAP operation. Controls and extensions allow you to specify additional information as part of a request or a response. Each extended operation is identified by an Object Identifier (OID), which is a string of octet digits that are required to add an attribute or objectclass of your own to an LDAP server. LDAP clients can send extended operation requests specifying the OID of the extended operation that should be performed and the data specific to that extended operation. When the LDAP server receives the request, it performs the extended operation and sends a response containing an OID and any additional data to the client.

For example, a client can include a control that specifies a sort with the search request that it sends to the server. When the server receives the search request, it sorts the search results before sending the search results back to the client. Servers can also send controls to clients. For example, a server can send a control with the authentication request that informs the client about password expiration.

By default, the OpenText eDirectory LDAP server loads all system extensions and selected optional extensions and controls when the LDAP server starts up. The extensionInfo attribute of LDAP Server object for optional extensions allows the system administrator to select or deselect the optional extensions and controls.

To enable extended operations, LDAP 3 protocol requires servers to provide a list of supported controls and extensions in the supportedControl attribute and supportedExtension attribute in the rootDSE. rootDSE (DSA [Directory System Agent] Specific Entry) is an entry that is located at the root of the Directory Information Tree (DIT). For more information, see “Getting Information about the LDAP Server” on page 409.

For a list of supported LDAP controls and extensions, see “LDAP Controls” (https://www.novell.com/documentation/developer/ldapover/ldap_enu/data/cchbehhc.html) and “LDAP Extensions” (https://www.novell.com/documentation/developer/ldapover/ldap_enu/data/a6ik7oi.html) in the LDAP and OpenText eDirectory Integration NDK.

Using LDAP Tools on Linux

OpenText eDirectory includes the following LDAP tools, stored in `/opt/novell/eDirectory/bin`, to help you manage the LDAP directory server.

NOTE: OpenText eDirectory 9.0 onwards, the `.PEM` certificates are passed through specific TLS variables. These variables can either be defined in the `/etc/opt/novell/eDirectory/conf/openldap/ldap.conf` file or can be exported individually. For more information, see OpenLdap Documentation Website and Man Pages.

Tool	Description
ice	Imports entries from a file to an LDAP directory, modifies the entries in a directory from a file, exports the entries to a file, and adds attribute and class definitions from a file.
ldapadd	Adds new entries to an LDAP directory.

Tool	Description
ldapdelete	Deletes entries from an LDAP directory server. The ldapdelete tool opens a connection to an LDAP server, binds, and deletes one or more entries.
ldapmodify	Opens a connection to an LDAP server, binds, and modifies or adds entries.
ldapmodrdn	Modifies the relative distinguished name (RDN) of entries in an LDAP directory server. Opens a connection to an LDAP server, binds, and modifies the RDN of entries.
ldapsearch	Searches entries in an LDAP directory server. Opens a connection to an LDAP server, binds, and performs a search using the specified filter. The filter should conform to the string representation for LDAP filters as defined in RFC 2254 (http://www.ietf.org/rfc/rfc2254.txt).
ndsindex	Creates, lists, suspends, resumes, or deletes indexes.

For more information, see "LDAP Tools" (https://www.novell.com/documentation/developer/cldap/) in the *LDAP Libraries for C Doc*.

To perform secure LDAP tools operations, refer to Ensuring Secure OpenText eDirectory Operations on Linux Computers and include the PEM file in all command line LDAP operations that establish secure LDAP connections to OpenText eDirectory.

LDAP Tools

The LDAP utilities can be used to delete entries, modify entries, add entries, extend the schema, modify relative distinguished names, move entries to new containers, create search indexes, or perform searches.

NOTE: In compliance with RFC 2256, the LDAP interface of OpenText eDirectory only allows binds to occur with passwords up to 128 characters in length. Also, passwords can only be set to have up to 128 characters when set through LDAP.

ldapadd

The ldapadd utility adds new entries. It has the following syntax:

```
ldapadd [-c] [-C] [-l] [-M] [-P] [-r] [-n] [-v] [-F] [-l limit] [-M[M]] [-
d debuglevel] [-D binddn] [[-W]| [-w passwd]] [-h ldaphost] [-p
ldapport] [-P version] [-Z[Z]] [-f file]
```

If the `-f` option is specified, ldapadd reads the modifications from a file. If the `-f` option is not specified, ldapadd reads the modifications from stdin.

TIP: Output from the LDAP utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file. For example, `ldapadd [options] > out.txt`.

Option	Description
`-a`	Adds new entries. The default for ldapmodify is to modify existing entries. If invoked as ldapadd, this flag is always set.
`-r`	Replaces existing values by default.
`-c`	Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error.
`-f file`	Reads the entry modification information from an LDIF file instead of from standard input. The maximum length of a record is 4096 lines.
`-F`	Forces the application of all changes regardless of the contents of input lines that begin with `replica:`. By default, `replica:` lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied.

Common Options for All LDAP Tools

There are some options that are common to all LDAP tools. These are listed in the following table:

Option	Description
`-C`	Enables referral following (anonymous bind).
`-d debuglevel`	Sets the LDAP debugging level to debuglevel. The ldapmodify tool must be compiled with LDAP_DEBUG defined for this option to have any effect.
`-D binddn`	Uses binddn to bind to the LDAP directory. *binddn* should be a string-represented DN as defined in RFC 1779.
`-f file`	Reads a series of lines from file, performing one LDAP search for each line. In this case, the filter given on the command line is treated as a pattern, where the first occurrence of %s is replaced with a line from the file. If the file is a single hyphen (-) character, then the lines are read from standard input.
`-h ldaphost`	Specifies an alternate host on which the LDAP server is running.
`-l limit`	Specifies the connection timeout (in seconds).
`-M`	Enables Manage DSA IT control (non-critical).
`-MM`	Enables Manage DSA IT control (critical).
`-n`	Shows what would be done, but does not actually modify entries. Useful for debugging in conjunction with -v.
`-p ldapport`	Specifies an alternate TCP™ port where the LDAP server is listening.
`-P version`	Specifies the LDAP version (2 or 3).
`-v`	Uses verbose mode with many diagnostics written to standard output.
`-w passwd`	Uses *passwd* as the password for simple authentication.
`-W`	Prompts for simple authentication. This option is used instead of specifying the password on the command line.

Option	Description
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file. This validates the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation, the operation is aborted. If a port is specified with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file. This validates the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

Examples

Assume that the file /tmp/entrymods exists and has the following contents:

```
dn: cn=Modify Me, o=University of Michigan, c=US
changetype: modify
replace: mail
mail: modme@terminator.rs.itd.umich.edu
-
add: title
title: Manager
-
add: jpegPhoto
jpegPhoto: /tmp/modme.jpeg
-
delete: description
-
```

In this case, the command ldapmodify -b -r -f /tmp/entrymods will replace the contents of the Modify Me entry's mail attribute with the value modme@terminator.rs.itd.umich.edu, add a title of Manager, add the contents of the file /tmp/modme.jpeg as a jpegPhoto, and completely remove the description attribute.

The same modifications as above can be performed using the older ldapmodify input format:

```
cn=Modify Me, o=University of Michigan, c=US

mail=modme@terminator.rs.itd.umich.edu

+title=Manager

+jpegPhoto=/tmp/modme.jpeg

-description
```

and the command:

```
ldapmodify -b -r -f /tmp/entrymods
```

Assume that the file `/tmp/newentry` exists and has the following contents:

```
dn: cn=Barbara Jensen, o=University of Michigan, c=US

objectClass: person

cn: Barbara Jensen

cn: B Jensen

sn: Jensen

title: Manager

mail: bjensen@terminator.rs.itd.umich.edu

uid: bjensen
```

In this case, the command `ldapadd -f /tmp/entrymods` will add a new entry for B Jensen, using the values from the file `/tmp/newentry`.

Assume that the file `/tmp/newentry` exists and has the following contents:

```
dn: cn=Barbara Jensen, o=University of Michigan, c=US

changetype: delete
```

In this case, the command `ldapmodify -f /tmp/entrymods` will remove B Jensen's entry.

ldapdelete

The ldapdelete utility deletes the specified entry. It opens a connection to an LDAP server, binds, and then deletes. It has the following syntax:

```
ldapdelete [-n] [-v] [-c] [-r] [-l] [-C] [-M] [-d debuglevel] [-f file] [-
D binddn] [[-W]| [-w passwd]] [-h ldaphost] [-p ldapport] [-Z[Z]] [dn]...
```

The dn parameter is a list of distinguished names of the entries to be deleted.

It interacts with the `-f` option in the following ways:

- If the `-f` option is missing from the command line, and DNs are specified on the command line, the utility deletes the specified entries.

- If both dn and the `-f` option are in the command line, the utility reads the file for the DNs to delete and ignores any DNs in the command line.
- If both dn and the `-f` option are missing in the command line, the utility reads the DN from stdin.

TIP: Output from the LDAP utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, `ldapdelete [options] > out.txt`.

Option	Description
`-c`	Continuous operation mode. Errors are reported, but ldapdelete will continue with deletions. The default is to exit after reporting an error.
`-f file`	Reads a series of lines from the file, performing one LDAP search for each line. In this case, the filter given on the command line is treated as a pattern, where the first occurrence of `%s` is replaced with a line from the file.
`-r`	Delete recursively.

NOTE: Refer to "Common Options for All LDAP Tools" on page 351 for more details on common options.

Example

The command `ldapdelete "cn=Delete Me, o=University of Michigan, c=US"` will attempt to delete the entry named with the commonName Delete Me directly below the University of Michigan organizational entry. In this case, it would be necessary to supply a `binddn` and `passwd` for the deletion to be allowed (see the `-D` and `-w` options).

ldapmodify

The ldapmodify utility modifies the attributes of an existing entry or adds new entries. It has the following syntax:

```
ldapmodify [-a] [-c] [-C] [-M] [-P] [-r] [-n] [-v] [-F] [-l limit] [-M[M]]
[-d debuglevel] [-D binddn] [[-W]|[-w passwd]] [-h ldaphost] [-p ldap-port]
[-P version] [-Z[Z]] [-f file]
```

If the `-f` option is specified, ldapmodify reads the modifications from a file. If the `-f` option is not specified, ldapmodify reads the modifications from stdin.

TIP: Output from the LDAP utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file. For example, `ldapmodify [options] > out.txt`.

Option	Description
`-a`	Adds new entries. The default for ldapmodify is to modify existing entries. If invoked as ldapadd, this flag is always set.
`-r`	Replaces existing values by default.
`-c`	Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error.
`-f` *`file`*	Reads the entry modification information from an LDIF file instead of from standard input. The maximum length of a record is 4096 lines.
`-F`	Forces the application of all changes regardless of the contents of input lines that begin with `replica:`. By default, `replica:` lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied.

NOTE: Refer to "Common Options for All LDAP Tools" on page 351 for more details on common options.

ldapmodrdn

The ldapmodrdn modifies the relative distinguished name of an entry. It can also move the entry to a new container. It has the following syntax:

```
ldapmodrdn [-r] [-n] [-v] [-c] [-C] [-l] [-M] [-s newsuperior] [-d
debuglevel] [-D binddn] [[-W]|[-w  passwd]]  [-h ldaphost] [-p ldapport] [-
Z[Z]] [-f file] [dn newrdn]
```

NOTE: Output from the LDAP utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file. For example, `ldapmodrdn [options] > out.txt`.

Option	Description
`-c`	Continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error.
`-f` *`file`*	Reads the entry modification information from the file instead of from standard input or the command line. Make sure that there are no blank lines between the old RDN and new RDN, or the `-f` option will fail.
`-r`	Removes old RDN values from the entry. The default is to keep old values.
`-s` *`newsuperior`*	Specifies the distinguished name of the container to which the entry is moving.

NOTE: Refer to Common Options for All LDAP Tools (page 351) for more details on common options.

Example

Assume that the file `/tmp/entrymods` exists and has the following contents:

```
cn=Modify Me, o=University of Michigan, c=US

cn=The New Me
```

ldapsearch

The ldapsearch utility searches the directory for specified attributes and object classes. It has the following syntax:

```
ldapsearch [-n] [-u] [-v] [-t] [-A] [-T] [-C] [-V] [-M] [-P] [-L] [-d
debuglevel] [-f file] [-D binddn] [[-W]| [-w bindpasswd]] [-h ldaphost] [-
p ldapport] [-b searchbase] [-s scope] [-a deref] [-l time limit] [-z size
limit] [-Z[Z]] filter [attrs....]
```

The ldapsearch tool opens a connection to an LDAP server, binds, and performs a search using the filter. The filter should conform to the string representation for LDAP filters as defined in RFC 2254 (http://www.ietf.org/rfc/rfc2254.txt).

If ldapsearch finds one or more entries, the attributes specified by `attrs` are retrieved and the entries and values are printed to standard output. If no attributes are listed, all attributes are returned.

TIP: Output from the LDAP utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file. For example, `ldapsearch [options] filter [attribute list] > out.txt`.

Option	Description
`-a` *`deref`*	Specifies how to handle the dereferencing of an alias. It uses the following values: ◆ Never: Aliases are never dereferenced while locating the base object or searching. ◆ Always: Aliases are always dereferenced when locating the base object and searching. ◆ Search: Aliases are dereferenced when searching subordinates of the base object but not when locating the base object. ◆ Find: Aliases are dereferenced when locating the base object but not when searching for the subordinates of the base object.
`-A`	Retrieves attributes only (no values). This is useful when you want to see if an attribute is present in an entry and when you are not interested in the specific values.
`-CC`	Enables referral following (authenticated bind with same bind DN and password).
`-b` *`searchbase`*	Use *`searchbase`* as the starting point for the search.
`-L`	Prints entries in the LDIF format.

Option	Description
`-LL`	Prints entries in the LDIF format without comments.
`-LLL`	Prints entries in the LDIF format without comments and version.
`-s scope`	Specifies the scope of the search. Scope should be `base`, `one`, or `sub` to specify a base object, one-level, or subtree search. The default is `sub`.
`-S attribute`	Sorts the entries returned, based on attribute. The default is not to sort entries returned. If an attribute is a zero-length string (" "), the entries are sorted by the components of their distinguished name. See ldap_sort for more details. ldapsearch normally prints out entries as it receives them. The use of the `-S` option defeats this behavior, causing all entries to be retrieved, sorted, and then printed.
`-t`	Writes retrieved binary values to a set of temporary files. This is useful for dealing with non-ASCII values such as jpegPhoto or audio.
`-tt`	Writes all values to temporary files.
`-T path`	Writes files to directory specified by path (default: `/tmp`).
`-u`	Includes the user-friendly form of the distinguished name (DN) in the output.
`-V`	URL prefix for files.
`-V prefix`	Specifies the URL prefix for files (default: `file://tmp/`).
`-z sizelimit`	Waits at most *`sizelimit`* entries for a search to complete.

NOTE: Refer to "Common Options for All LDAP Tools" on page 351 for more details on common options.

Examples

The following command:

```
ldapsearch "cn=mark smith" cn telephoneNumber
```

will perform a subtree search (using the default search base) for entries with a commonName of `mark smith`. The commonName and telephoneNumber values will be retrieved and printed to standard output. The output might look like the following if two entries are found:

```
cn=Mark D Smith, ou="College of Literature, Science, and the Arts",
ou=Students, ou=People, o=University of Michigan, c=US

cn=Mark Smith

cn=Mark David Smith

cn=Mark D Smith 1

cn=Mark D Smith

telephoneNumber=+1 313 930-9489
```

```
cn=Mark C Smith, ou=Information Technology Division, ou=Faculty and Staff,
ou=People,o=University of Michigan, c=US

cn=Mark Smith

cn=Mark C Smith 1

cn=Mark C Smith

telephoneNumber=+1 313 764-2277
```

The command:

```
ldapsearch -u -t "uid=mcs" jpegPhoto audio
```

will perform a subtree search using the default search base for entries with user IDs of mcs. The user-friendly form of the entry's DN will be output after the line that contains the DN itself, and the jpegPhoto and audio values will be retrieved and written to temporary files. The output might look like the following if one entry with one value for each of the requested attributes is found:

```
cn=Mark C Smith, ou=Information Technology Division, ou=Faculty and Staff,
ou=People, o=University of Michigan, c=US

Mark C Smith, Information Technology Division, Faculty and Staff, People,
University of Michigan, US

audio=/tmp/ldapsearch-audio-a19924

jpegPhoto=/tmp/ldapsearch-jpegPhoto-a19924
```

The following command will perform a one-level search at the c=US level for all organizations whose organizationName begins with university.:

```
ldapsearch -L -s one -b "c=US" "o=university*" o description
```

Search results will be displayed in the LDIF format. The organizationName and description attribute values will be retrieved and printed to standard output, resulting in output similar to the following:

```
dn: o=University of Alaska Fairbanks, c=US

o: University of Alaska Fairbanks

description: Preparing Alaska for a brave new yesterday.

description: leaf node only

dn: o=University of Colorado at Boulder, c=US

o: University of Colorado at Boulder

description: No personnel information

description: Institution of education and research

dn: o=University of Colorado at Denver, c=US

o: University of Colorado at D
```

ndsindex

The ndsindex utility creates, lists, suspends, resumes, or deletes indexes and compound indexes. You can specify multiple attributes separated by $ sign in the ndsindex utility for compound index. It has the following syntax:

NOTE: ◆You can specify multiple attributes for compound index. OpenText recommends you to enter up to 3 attributes for better performance. In case of value type compound index, you can add maximum 5 attributes.

- We recommend you to connect ndsindex utility to the same server where the index has been added.

```
ndsindex list [-h <hostname>] [-p <port>] -D <bind DN> -W|[-w <password>]
[-l limit] -s <eDirectory Server DN> [-Z[Z]] [<indexName1>,
<indexName2>.....]
```

```
ndsindex add -a [-h <hostname>] [-p <port>] -D <bind DN> -W|[-w <password>]
[-l limit] -s <eDirectory Server DN> [-Z[Z]] <indexDefinintion1>
[<indexDefinintion2>.....]
```

NOTE: ◆Using the -a option, you can prefix ancestor ID attribute to the list of attributes passed while creating a new index. An index with ancestor id can only be created with value index type. Presence and Substring index types are not supported with ancestor id.

- Database size increases after creating index with ancestor id.

```
ndsindex delete [-h <hostname>] [-p <port>] -D <bind DN> -W|[-w <password>]
[-l limit] -s <eDirectory Server DN> [-Z[Z]] <indexName1>
[<indexName2>.....]
```

```
ndsindex resume [-h <hostname>] [-p <port>] -D <bind DN> -W|[-w <password>]
[-l limit] -s <eDirectory Server DN> [-Z[Z]] <indexName1>
[<indexName2>.....]
```

```
ndsindex suspend [-h <hostname>] [-p <port>] -D <bind DN> -W|[-w
<password>] [-l limit] -s <eDirectory Server DN> [-Z[Z]] <indexName1>
[<indexName2>.....]
```

Option	Description
list	Lists the specified indexes. If the index is not specified, ndsindex lists all existing indexes on the server.
add	Creates new indexes.
delete	Deletes the specified indexes.
resume	Resumes the specified indexes from an off-line state.
suspend	Suspends the specified indexes to an off-line state.
-s *eDirectory Server DN*	Specifies the eDirectory Server DN.

NOTE: Refer to "Common Options for All LDAP Tools" on page 351 for more details on common options.

Examples

To list the indexes on the server MyHost, enter the following command:

```
ndsindex list -h MyHost -D cn=admin,o=mycompany -w password -s
cn=MyHost,o=novell
```

To create a substring index with the name MyIndex on the email address attribute, enter the following command:

```
ndsindex add -h myhost -D cn=admin, o=mycompany -w password -s cn=myhost,
o=novell "MyIndex;email address;substring"
```

To create a value index with the name MyIndex on the city attribute with an ancestor ID, enter the following command:

```
ndsindex add -a -h myhost -D cn=admin,o=mycompany -w password -s
cn=myhost,o=novell "MyIndex;city;value"
```

To create a presence index with the name MyIndex on the homephone attribute, enter the following command:

```
ndsindex add -h myhost -D cn=admin,o=mycompany -w password -s
cn=myhost,o=novell "MyIndex;homephone;presence"
```

To delete the index named MyIndex, enter the following command:

```
ndsindex delete -h myhost -D cn=admin,o=mycompany -w password -s
cn=myhost,o=novell MyIndex
```

To suspend the index named MyIndex, enter the following command:

```
ndsindex suspend -h myhost -D cn=admin,o=mycompany -w password -s
cn=myhost,o=novell MyIndex
```

To resume the index named MyIndex, enter the following command:

```
ndsindex resume -h myhost -D cn=admin,o=mycompany -w password -s
cn=myhost,o=novell MyIndex
```

Examples for Compound Indexes

To create a value index with the name `MyIndex` on the `email address` and `surname` attribute, enter the following command:

```
ndsindex add –h myhost –D cn=admin, o=mycompany –w password –s cn=myhost,
o=netiq 'MyIndex;email address$surname;value'
```

NOTE: You cannot create a compound index for `Presence` and `Substring` type.

Extensible Match Search Filter

The LDAP 3 core protocol specification defined in RFC 2251 (http://www.ietf.org/rfc/rfc2251.txt) requires LDAP servers to recognize a search element called an extensible match filter. An extensible match allows an LDAP client to specify the following items in a search filter:

- An optional attribute name
- An optional matching rule
- A flag to indicate if the DN attributes should be considered a part of the entry
- The value to be used for the match

The following is the string representation of the extensible match search filter:

```
extensible = attr [":dn"] [":" matchingrule] ":=" value /
             [:dn"] ":" matchingrule ":=" value
```

The following table lists the Extensible Match search filter parameters:

Parameter	Description
attr	Specifies the attribute to match on.
[":*dn*"]	Indicates that the matching rule should be included in the comparison match.
[":" *matchingrule*]	Designates the matching rule to be used.
":="	Without a matching rule results in an equality match.
value	Comparison value

The extensibleMatch is a new filter provided in LDAP 3. If the matchingRule field is absent, the attribute field MUST be present, and the equality match is performed for that attribute. If the attribute field is absent and matchingRule is present, the matchValue is compared against all attributes in an entry that supports that matchingRule, and the matchingRule determines the syntax for the assertion value.

The filter item evaluates as

- TRUE if it matches with at least one attribute in the entry.
- FALSE if it does not match any attribute in the entry.
- Undefined if the matchingRule is not recognized or the assertionValue cannot be parsed.

If the type field along with the matchingRule is present, the matchingRule must be one permitted for use with that type, otherwise the filter item is undefined. If the :dn is specified in the search filter, the match is applied against all the attributes in an entry's distinguished name as well, and also evaluates to TRUE if there is at least one attribute in the distinguished name for which the filter item evaluates to TRUE. The dnAttributes field is present so that there does not need to be multiple versions of generic matching rules such as for word matching, one to apply to entries and another to apply to entries and DN attributes as well.

Essentially, an extensible match filter allows an LDAP client to achieve two objectives:

- Support multiple matching rules for same type of data

- Include DN elements in the search criteria

  The DN specification allows matching on specific elements of the DN.

OpenText eDirectory 8.7.3 and later versions support the extensible match filter for matching on the DN attributes. The other elements of the extensible match search filter, namely the matching rule, are treated as undefined and ignored. The DN matching allows an LDAP client to drastically reduce the searches required to locate an object in an OpenText eDirectory tree. For example, a complex LDAP search filter such as

```
(&(ou:dn:=sales)(objectclass=user))
```

would let you have a listing of all the User objects in the sales function (that is, anywhere under the sales containers).

Usage Examples

The following are examples of the string representations of extensible match search filter that are supported in OpenText eDirectory 8.7.3 and later versions.

```
(o:dn:=Ace Industry)
```

This example illustrates the use of the `:dn` notation. The attributes of an entry's distinguished name should be considered part of the entry when evaluating the match. It denotes an equality match.

```
(:dn:2.4.8.10:=Dino)
```

This example is a filter that should be applied to any attribute of an entry. Attributes contained in the DN with the matching rule 2.4.8.10 should also be considered.

The following are some examples of the string representation of extensible match search filter that are *not* supported in OpenText eDirectory 8.7.3 and later versions:

```
(cn:1.2.3.4.5:=John Smith)
```

This example illustrates a filter that specifies the attributes type cn and value John Smith. It mandates that the match should be performed by the directory server according to the matching rule identified by the oid 1.2.3.4.5.

```
(sn:dn:2.4.6.8.10:=Barbara Jones)
```

This example illustrates the use of the `:dn` notation to indicate that matching rule 2.4.6.8.10 should be used when making comparisons, and that the attributes of an entry's distinguished name should be considered part of the entry when evaluating the match.

LDAP Transactions

OpenText eDirectory LDAP server supports clubbing of multiple update operations into a single atomic operation - also called a transaction. The support for transactions over LDAP in OpenText eDirectory is based on two Internet specifications – "LDAP Transactions" (http://www.watersprings.org/pub/id/draft-zeilenga-ldap-txn-05.txt) and "LDAP: Grouping of Related Operations" (http://www.watersprings.org/pub/id/draft-zeilenga-ldap-grouping-05.txt).

LDAP transactions allow an LDAP application to send several LDAP update operations (add, modify, delete, rename) as a group and then commit or abort this whole group of operations.

There are few entities which figure in the context of LDAP transactions:

- CreateGroupingRequest (2.16.840.1.113719.1.27.103.1) – This is LDAP extended operation which allows grouping of related operations. The extended operation carries a value – createGroupType which identifies the type of grouping requested. For LDAP transactions, the grouping type is transactionGroupingType. (2.16.840.1.113719.1.27.103.8)
- CreateGroupingResponse (2.16.840.1.113719.1.27.103.1) – This is the response of the LDAP server to the createGroupingRequest and contains 2 response fields – groupCookie and an optional createGroupValue.
- GroupingControl (2.16.840.1.113719.1.27.103.7) - This is used to indicate association of an operation to a grouping via the groupCookie which is the value carried by this control.
- EndGroupingRequest (2.16.840.1.113719.1.27.103.2) – This is another LDAP extended operation used to indicate the end of a grouping Request. In case of LDAP transactions, this indicates the settling of the transaction – resulting in a commit or an abort of the transaction.
- EndGroupingResponse (2.16.840.1.113719.1.27.103.2) – This is the response of the LDAP server to the endGroupingResponse indicating either success or otherwise to the LDAP client.

Following is the sequence of requests and responses exchanged between the LDAP server and the LDAP client in an LDAP transaction:

- If a client wants to send a number of LDAP operations to be processed by a server in an atomic operation, i.e., transaction, it should first send a createGroupingRequest, with a createGroupType of transactionGroupingType and no createGroupValue.
- If the OpenText eDirectory server is capable of handling transactions, it sends back a success result code, with a groupingCookie, which uniquely identifies the grouping requested by the client. Otherwise, the server shall return a non-successful result code indicating the reason for the failure to the client.
- If the client receives success result code from the server, it then attaches a GroupingControl, which includes the groupingCookie returned by the server, to subsequent update operations to indicate that they are to be processed as part of a single transaction. If the server is willing and able to process the update operation as part of the transaction, the server shall return success and put this request in a queue. If the server is unwilling or unable to process the update operation as part of the transaction, the server shall return a non-successful result code indicating the reason for the failure to the client.
- After the client has sent all the update operations accompanied by the grouping control to the server, the client sends an endGroupingRequest with the groupingCookie to the server to indicate that it wants to settle the transaction. The absence of endGroupValue indicates a commit request where as presence of an empty endGroupValue indicates abort request.
- The server applies all the pending operations in one transaction. If it succeeds, it shall return success. Otherwise, it shall return a non-successful result code.
- If at any time during the above exchange between the client and server, the server is unwilling or unable to continue the specification of a transaction, the server issues an endGroupingNotice (2.16.840.1.113719.1.27.103.4). Subsequent use of cookie by the client shall result in a response containing a non-success result code.

The support for LDAP transactions is indicated by the presence of the transactionGroupingType in the supportedGroupingTypes attribute of the rootDSE entry.

The LDAP transaction implementation in OpenText eDirectory is based on a dated version of the LDAP transaction specification. The latest revision of the LDAP transactions draft as of this writing is available at "Lightweight Directory Access Protocol (LDAP) Transactions" (http://tools.ietf.org/html/rfc5805).

Limitations

The LDAP transactions feature has the following limitations:

- All the objects affected by the operations grouped as a transaction need to be hosted locally on the server. None of these operations should require the LDAP server to chain to another server.
- Schema modifications and Modify DN operation (Subtree move?) is not allowed to be grouped in an LDAP transaction.
- Passwords and attributes with stream syntax cannot be added as part of an LDAP transaction.
- Nesting of one transaction within another is not supported.

14 Configuring LDAP Services for OpenText eDirectory

The OpenText installation program automatically installs LDAP Services for OpenText eDirectory. For information on installing OpenText eDirectory, see the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

This chapter explains the following:

- "Loading and Unloading LDAP Services for OpenText eDirectory" on page 365
- "Verifying That the LDAP Server Is Loaded" on page 366
- "Verifying That the LDAP Server Is Running" on page 367
- "Preventing POODLE Attack by Disabling SSLv3" on page 368
- "Configuring LDAP Objects" on page 369
- "Refreshing the LDAP Server" on page 383
- "Authentication and Security" on page 384
- "Using the LDAP Server to Search the Directory" on page 393
- "Configuring for Superior Referrals" on page 403
- "Persistent Search: Configuring for OpenText eDirectory Events" on page 407
- "Getting Information about the LDAP Server" on page 409
- "Configuring Generalized Time Support" on page 411
- "Configuring Permissive Modify" on page 411
- "Proxied Authorization Control" on page 412
- "LDAP Paged Search Control" on page 412
- "LDAP Extended DN Control" on page 413
- "Auditing LDAP Events" on page 416

For information on LDAP tools, see the LDAP Tools NDK (http://developer.novell.com/documentation/cldap/ltoolenu/data/hevgtl7k.html).

Loading and Unloading LDAP Services for OpenText eDirectory

To load LDAP Services for OpenText eDirectory, enter the following commands:

Server	Command
Windows	In the DHost (NDSCONS) screen, click **nldap.dlm** > **Start**.

Server	Command
Linux	At the Linux prompt, enter: `/opt/novell/eDirectory/sbin/nldap -l`

To unload LDAP Services for OpenText eDirectory, enter the following commands:

Server	Command
Windows	In the DHost (NDSCONS) screen, click **nldap.dlm** > **Stop**.
Linux	In the DHost remote management page, to unload LDAP, click the *LDAP v3 for OpenText eDirectory* action icon to stop. or At the Linux prompt, enter: `/opt/novell/eDirectory/sbin/nldap -u`

Verifying That the LDAP Server Is Loaded

Before configuring LDAP objects, verify that the LDAP server is loaded and functional. This section explains how to verify that the LDAP server is loaded. To verify that the server is running and functional, see "Verifying That the LDAP Server Is Running" on page 367.

On Windows

1. On a Windows server, open `ndscons.exe`.

   Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**.
2. On the **Services** tab, scroll to **nldap.dlm**, then view the **Status** column.

   The column displays Running.

On Linux

To verify if the LDAP Server is running, run the following command:

```
ndstrace -c modules | grep nldap
```

If the LDAP server is not loaded or running, an error appears stating that the nldap module is not loaded.

You can also use the following options:

- To check if LDAP server is running and listening on the SSL port, run the `nldap -s` command.
- To check if LDAP server is running and listening on the TCL port, run the `nldap -c` command.

These will list all the instances of OpenText eDirectory running with out any error. If the LDAP server is not loaded and not listening on either of the ports, the above commands display the error -255 (ensure that LDAP Server is running).

Verifying That the LDAP Server Is Running

After the LDAP server is loaded, verify that it is running. Then verify that a device is listening.

- "Scenarios" on page 367
- "Verifying That The LDAP Server Is Running" on page 368
- "Verifying That A Device Is Listening" on page 368

Scenarios

Typically, the LDAP server runs as soon as it is loaded. However, either of two scenarios can prevent the server from running properly.

Scenario: The Server Is in a Zombie State. The LDAP server loads as long as the DHost Loaders can resolve external dependencies. However, the LDAP server doesn't run properly until it can get a valid configuration from the two configuration objects (the LDAP Server and LDAP Group objects).

While the LDAP server is in a loaded-but-not-running (zombie) state, it periodically tries to find and read the configuration objects. If the objects are misconfigured or corrupted, the LDAP server stays in the zombie state until the server (`nldap.nlm`, `nldap.dlm`, `libnldap.so`, or `libnldap.sl`) is unloaded or taken down.

The Loaders show that the LDAP server is loaded, but no LDAP ports (389, 636) are opened by `nldap.nlm` (or `nldap.dlm`, `libnldap.so`, or `libnldap.sl`). Also, no LDAP client requests are serviced.

DSTrace messages will show the periodic attempts and the reason why the server cannot come up to the running state.

Scenario: Denial of Service . At Digital Airlines, the server is processing a very long (20 minutes or more) search operation. The search is, in effect, looking for a needle in a haystack.

During this search, Henri does one of the following:

- Changes a configuration parameter and updates a configuration object.
- Clicks **Refresh Server Now**.
- Unloads the LDAP server (`nldap.nlm`, `nldap.dlm`, `libnldap.so`, or `libnldap.sl`).
- Tries to take the entire server down.

The LDAP server waits until all current operations complete before applying any new update. The server also postpones new operations from running until the update is complete. This delay can cause the server to appear to stop responding to new requests until the search is done and the server can refresh itself. Or the server appears to hang during the unload.

If the search request is long but has many hits, and Henri unloads the LDAP server, it aborts the search and quickly unloads when the next hit is returned to the client. However, if the search request has only one or no hits in 20 minutes, the LDAP server isn't able to abandon the NDS® or OpenText eDirectory request in progress.

For a refresh or update, the search will not be aborted even if it has many hits to return to the client.

Verifying That The LDAP Server Is Running

To verify that the LDAP service is running, use the OpenText Import Conversion Export Utility (ICE). At a workstation, run `ice.exe` or use OpenText Identity Console.

If you enter an IP address and a port number and then get a connection, the server is functional. Otherwise, you receive an error message. Download (view) either the log file or the export file.

Verifying That A Device Is Listening

Verify that a device is listening on port 389.

1. At the command line, enter

   ```
   netstat -a
   ```

2. Find a line where the local address is *`servername`*`:389` and the state is `LISTENING`.

If one of the following situations occurs, run OpenText iMonitor:

- You are unable to get information from the ICE utility
- You are uncertain that the LDAP server is handling LDAP requests

For information on OpenText iMonitor, see "Configuration Files" on page 225 and "Configuring Trace Settings" on page 233.

For information on LDAP requests, see "Communicating with OpenText eDirectory through LDAP" in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

Preventing POODLE Attack by Disabling SSLv3

If your OpenText eDirectory uses LDAPS protocol with SSLv3 for a secure communication, be aware that SSLv3 is vulnerable to POODLE attack as per CVE-2014-3566.

By default, OpenText eDirectory runs in FIPS mode and does not allow communication over SSLv3. See Configuring eDirectory in FIPS Mode for more information. If you disable FIPS mode for TLS on your OpenText eDirectory server, you may want to disable SSLv3 for LDAP using the following procedure:

Workaround:

1. Download and install the latest OpenText Identity Console for OpenText eDirectory from the Software License and Download portal.
2. Launch OpenText Identity Console.
3. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
4. Select **LDAP Server**.
5. On the **Modify LDAP Server** page > **Configuration** drop down > select **SSLv3** check box.
6. Click **Save**.

NOTE: In a non-English environment, you cannot access the **SSLv3** option. To access this option, change the preferred display language to English.

---

7 Unload and load the LDAP Services for OpenText eDirectory.

For more information, see "Loading and Unloading LDAP Services for OpenText eDirectory" on page 365.

Configuring LDAP Objects

An OpenText eDirectory installation creates an LDAP server object and an LDAP Group object. The default configuration for LDAP Services is located in the directory on these two objects. You can modify the default configuration by using the LDAP Management task in OpenText Identity Console.

The LDAP server object represents server-specific configuration data.

The LDAP Group object contains configuration information that can be conveniently shared among multiple LDAP servers. This object provides common configuration data and represents a group of LDAP servers. The servers have common data.

You can associate multiple LDAP server objects with one LDAP Group object. All the associated LDAP servers then get their server-specific configuration from their LDAP server object but get common or shared information from the LDAP Group object.

By default, the OpenText eDirectory installation program installs a single LDAP Group object and a single LDAP server object for each `nldap.nlm` or `nldap.dlm`. Later, you can associate multiple LDAP server objects with a single LDAP Group object.

IMPORTANT: Although it is possible to associate newer versions of an LDAP server object with older versions of LDAP Group objects, we recommend that you don't mix versions. For example, avoid associating an LDAP Group object in eDirectory 8.7.3 SP9 with an LDAP server object in OpenText eDirectory 9.0 or later.

The amount of common information held in an LDAP Group object is limited. LDAP doesn't need to read many attributes because the data contained in the attributes is incredibly common. Many LDAP servers will need to use the same data. Without a common or shared Group object, you would have to replicate that data across each LDAP server.

The LDAP server object allows more server-specific configuration options and data than the LDAP Group object allows.

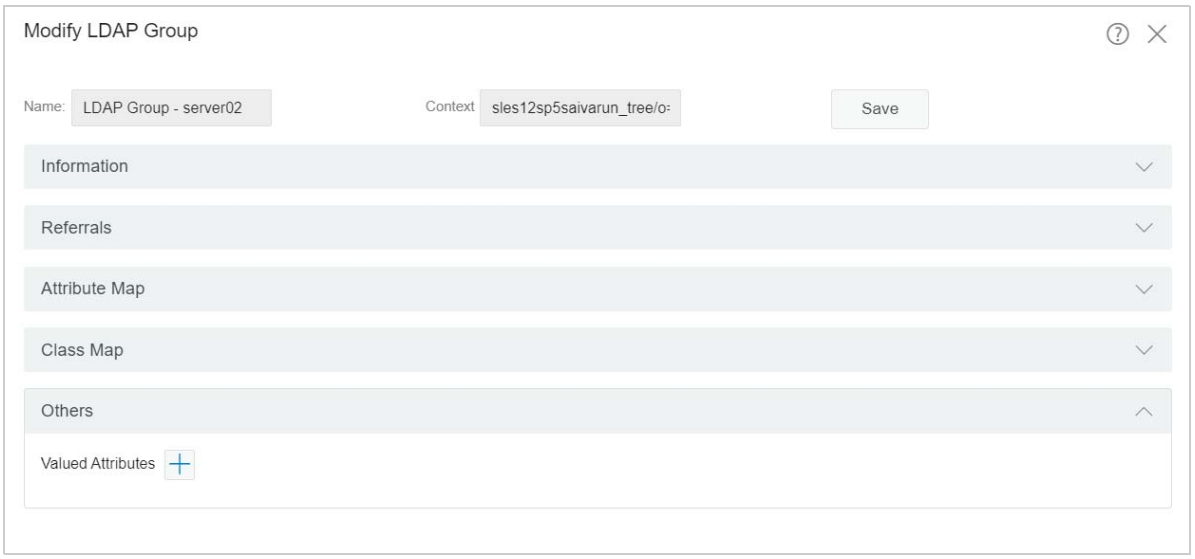
Both objects have DN-syntax attributes that point to each other.

An additional association must be made so that the LDAP server can find its configuration data. This association is through the NCP™ server, which holds the customary OpenText eDirectory configuration data. The OpenText eDirectory installation program automatically makes the association.

Every OpenText eDirectory server has an NCP Server object. The following figure illustrates the objects as displayed in OpenText Identity Console:



This object has an LDAP Server attribute, which points to the LDAP server object for a particular host OpenText eDirectory server. The following figure illustrates this attribute:



Typically, the LDAP server object, the LDAP Group object, and the NCP Server object are located in the same container. You name this container during the OpenText eDirectory installation, when you name the server and Admin context.

If you move the LDAP server object, you must place it in a writable replica.

Configuring LDAP Server and LDAP Group Objects on Linux

The LDAP configuration utility is ldapconfig. You can use ldapconfig on Linux, systems to modify, view, and refresh the attributes of LDAP server and LDAP Group objects.

Use the following syntax to view LDAP attribute values on Linux, systems:

```
ldapconfig get [...] | set attribute-value-list  [-t treename | -p
hostname[:port]] [-w password] [-a user FDN] [-f]
```

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w password] [-a user FDN]
[-V] [-R] [-H] [-f] -v attribute,attribute2...
```

Use the following syntax to modify values of LDAP attributes on Linux:

```
ldapconfig [-t tree_name | -p host_name[:port]] [-w password] [-a
admin_FDN] -s attribute=value,...
```

Parameter	Description
`-t treename`	Name of the OpenText eDirectory tree where the component will be installed.
`-p hostname`	The name of the host. You could specify the DNS name or IP address also.
`-w`	The password of the user having administration rights.
`-a`	The fully distinguished name of the user having administration rights. For example: cn=user.o=org1
`get | -V`	Lets you view all LDAP server/group attributes.
`get | -v attribute list`	Displays the current values of the attributes in the attribute list.
`set | -s attribute-value pairs`	Sets the attributes to the specified values.
`-v`	Lets you view the value of the LDAP attribute.
`-s`	Sets a value for an attribute of the installed components.
`-R`	Refreshes the LDAP server.
`-V`	Lets you view the current LDAP configuration settings.
`-H`	Lets you view the usage and help strings.
`-f`	Allows operations on a filtered replica.
attribute	A configurable LDAP server or group attribute name. For more information, see "Attributes on the LDAP Server Object" on page 371 and "Attributes on the LDAP Group Object" on page 380.

Examples

To view the value of the attribute in the attribute list, enter the following command:

```
ldapconfig [-t tree_name | -p host_name[:port]]
[-w password] [-a user_FDN] -v "Require TLS for simple binds with
password","searchTimeLimit"
```

To configure the LDAP TCP port number and search size limit to 1000, enter the following command:

```
ldapconfig [-t tree_name | -p host_name[:port]]
[-w password] [-a admin_FDN] -s "LDAP TCP Port=389","searchSizeLimit=1000"
```

Attributes on the LDAP Server Object

Use the LDAP server object to set up and manage the OpenText LDAP server properties.

The following table provides a description of the LDAP server attributes:

Attribute	Description
LDAP Server	The fully distinguished name of the LDAP server object in OpenText eDirectory.
LDAP Host Server	The fully distinguished name of the host OpenText eDirectory server that the LDAP server runs on.
LDAP Group	The LDAP Group object in OpenText eDirectory that this LDAP server is a member of.
LDAP Server Bind Limit	The number of clients that can simultaneously bind to the LDAP server. A value of 0 (zero) indicates no limit.
LDAP Server Idle Timeout	The period of inactivity from a client after which LDAP server terminates the connection with this client. A value of 0 (zero) indicates no limit.
LDAP Enable TCP	This option is deprecated. It is available through ldapInterfaces. For more information, see “ldapInterfaces” on page 376.
LDAP Enable TLS	This option has been deprecated. However, it is available through ldapInterfaces. For more information, see “ldapInterfaces” on page 376.
LDAP TCP Port	This option has been deprecated. However, it is available through ldapInterfaces. For more information, see “ldapInterfaces” on page 376.
LDAP TLS Port	This option has been deprecated. However, it is available through ldapInterfaces. For more information, see “ldapInterfaces” on page 376.
keyMaterialName	The name of the Certificate object in OpenText eDirectory that is associated with this LDAP server and will be used for SSL LDAP connections.
searchSizeLimit	The maximum number of entries that the LDAP server will return to an LDAP client in response to a search. A value of 0 (zero) indicates no limit. If the user has the administrator rights on the LDAP server object, the searchSizeLimit value is not considered. Any changes made to the administrative rights for a user will not be effective immediately because the administrative rights are cached. The changes to the administrative rights will be effective with the next LDAP server refresh. By default, the LDAP server refreshes once in every 30 minutes
searchTimeLimit	The maximum number of seconds after which an LDAP search will be timed out by the LDAP server. A value of 0 (zero) indicates no limit. If the user has the administrator rights on the LDAP server object, the searchTimeLimit value is not considered. Any changes made to the administrative rights for a user will not be effective immediately because the administrative rights are cached. The changes to the administrative rights will be effective with the next LDAP server refresh. By default, the LDAP server refreshes once in every 30 minutes.

Attribute	Description
filteredReplicaUsage	Specifies whether the LDAP server should use a filtered replica for an LDAP search. Values=1 (use filtered replica), 0 (do not use filtered replica)
sslEnableMutualAuthentication	Specifies whether SSL-based mutual authentication (Certificate-based client authentication) is enabled on the LDAP server.
ldapTLSVerifyClientCertificate	Enables or disables verification of the client certificate for a TLS operation through LDAP.
ldapNonStdAllUserAttrsMode	Enables or disables the non standard, all user, and operational attributes.

Attribute	Description
ldapBindRestrictions	Enables LDAP bind restrictions and cipher level on LDAP client connections.This attribute can be used to control client connections. You can set any of the following seven LDAP bind restrictions using OpenText Identity Console: • **NONE:** This is enabled by default. This option enables both anonymous simple bind and non-anonymous simple bind. The value of this option is 0. • **Disallow anonymous simple bind:** Set the value to 1 to disable the anonymous simple bind. Non-anonymous simple bind will be enabled. • **Disallow non-anonymous simple bind:** Set the value to 2 to disable non-anonymous simple bind. • **Disallow anonymous simple bind and non-anonymous simple bind:** Set the value to 3 to disable anonymous simple bind and non-anonymous simple bind. **NOTE:** Disabling non-anonymous simple bind will enforce appropriate grace login limits. • **Disallow unauthenticated bind:** Set the value to 4 to disable simple bind with no password. • **Disallow anonymous and unauthenticated bind:** Set the value to 5 to disable anonymous simple bind and unauthenticated bind. • **Disallow non-anonymous simple bind and unauthenticated bind:** Set the value to 6 to disable non-anonymous simple bind and unauthenticated bind. Anonymous simple bind will be enabled in this scenario. • **Disallow anonymous simple bind, non-anonymous simple bind and unauthenticated bind:** Set the value to 7 to disable anonymous simple bind, non-anonymous simple bind and unauthenticated bind. **NOTE:** The value from 4 to 7 can be set from the `ldapconfig` utility. OpenText Identity Console doesn't allow to set this value. For more information, see Table 14-1. For RSA and Elliptic Curve Digital Signature (ECDSA) algorithms, OpenText eDirectory allows you to use the following values to restrict the cipher usage: • **RSA:** Use the following values: ◦ **High Cipher (greater than 128-bit):** Set the value to 48 to specify the use of a cipher level larger than 128-bit encryption and some cipher suites with 128-bit keys. ◦ **Medium Cipher:** Set the value to 32 to specify the use of cipher level of 128-bit encryption. ◦ **Low Cipher:** Set the value to 16 to specify the use of 64 or 56-bit encryption excluding export cipher suites. ◦ **Export:** Specifies the use of a cipher level including 40 and 56-bit encryption.Value 0. The default is **High** with a cipher level larger than 128-bit encryption. If this value is set to **0**, after upgrading to OpenText eDirectory 9.1 SP4, the value will automatically be changed to **High**. **NOTE:** If FIPS mode is enabled for TLS, OpenText eDirectory ignores the cipher configuration and allows only **High** ciphers.

Attribute	Description
	Suite B Mode: Use the following values: ◆ **Suite B Cipher (128-bit):** Set value to 64 to enable Suite B mode operation by using 128-bit level of security. When you select this option, OpenText eDirectory permits both 128-bit and 192-bit level of security by peers (any LDAP clients). You can use either ECDSA 256 or ECDSA 384 certificate with this option. ◆ **Use Suite B Cipher (128-bit only):** Set value to 80 to enable Suite B mode operation by using 128-bit level of security. When you select this option, OpenText eDirectory does not allow 192-bit level of security by peers (any LDAP clients). You can only use ECDSA 256 certificate with this option. ◆ **Use Suite B Cipher (192-bit):** Set value to 96 to enable Suite B mode operation by using 192-bit level of security. When you select this option, OpenText eDirectory permits only 192-bit level of security by peers (any LDAP clients). You can only use ECDSA 384 certificate with this option. OpenText eDirectory allows you to use combination values of `ldapbindrestrictions` and cipher levels. For more information, see Table 14-1.
ldapChainSecureRequired	This is a boolean attribute. If enabled, chaining to other OpenText eDirectory will be over secure NCP. By default, ldapChainSecureRequired is disabled.

Attribute	Description

ldapInterfaces	A multi-valued SYN_CI_STRING attribute used to store LDAP URLs on which LDAP server listens (on both cleartext and secure ports). This attribute is useful in configuring multiple instances that require each instance of the OpenText eDirectory server to listen on a specific interface. It can be configured with the IP addresses and port numbers in the LDAP URL format.The LDAP server listens on these IP addresses and ports.
	
	The following are examples for IPv4 and IPv6 listeners.
	
	`ldap://192.168.1.1:389 - To specify for IPv4 specific address on clear text port`
	
	`ldaps://192.168.2.1:636 - To specify for IPv4 specific address on secure port`
	
	`ldap://[2015::3]:389 - To specify for IPv6 specific address on clear text port`
	
	`ldaps://[2015::3]:636 - To specify for IPv6 specific address on secure port`
	
	`ldap://[::]:389 - To specify for IPv6 unspecified address on clear text port`
	
	`ldaps://[::]:636 - To specify for IPv6 unspecified address on secure port`
	
	The LDAP Enable TCP, LDAP Enable TLS, LDAP TCP Port, and LDAP TLS Port attributes are not populated if a new server is configured from OpenText eDirectory 9.1 onwards. The `ldapInterface` atrribute values corresponding to the ports selected for ldap and ldaps during configuration are populated. For example, ldap://:389, ldaps://:636. By default, only IPv4 interface values are added to the ldapInterfaces attribute.
	
	During upgrade, OpenText eDirectory is triggered to delete the LDAP Enable TCP, LDAP Enable TLS, LDAP TCP Port, LDAP TLS Port attributes. It populates corresponding values of these attributes in `ldapInterface`. The `ldapconfig set` command takes comma separated values and replaces all the existing values with the new values.

ldapStdCompliance	OpenText eDirectory LDAP server by default does not return the sub-ordinate referrals for ONE level search. To enable this, you need to turn on `ldapStdCompliance` with a value 1. Setting this value will make the LDAP server return the sub-ordinate referrals for ONE level search.
ldapChainSecureRequired	This is a boolean attribute. If this is enabled, the chaining to other OpenText eDirectory will be over secure NCP. By default, the attribute will be disabled.
ldapEnablePSearch	Specifies whether or not the persistent search feature is enabled on the LDAP server. Values= yes, no
ldapMaximumPSearchOperations	An integer value that limits the number of concurrent persistent search operations possible. A value of 0 specifies unlimited search operations.

Attribute	Description
ldapIgnorePSearchLimitsForEvents	Indicates whether size and time limits should be ignored after the persistent search request has sent the initial result set. Values= yes, no If this attribute is set to false, the entire persistent search operation is subject to the search limits. If either limit is reached, the search fails with the appropriate error message.
ldapGeneralizedTime	Enable Generalized Time to display time in the *YYYYMMDDHHmmSS.0Z* format. Values= yes, no
ldapPermissiveModify	Enable Permissive Modify Control to extend the LDAP modify operation. If an attempt is made to delete an attribute that does not exist or to add any value to an attribute that already exists, the operation goes through without displaying any error message Values= yes, no
ldapSSLConfig	This attribute allows you to define the TLS protocols and Ciphers in the LDAP server object. By default, this attribute is disabled. This configuration attribute follows the following order of precedence: ◆ Presence of `ldapSSLConfig` attribute value on the LDAP server object ◆ Presence of `ldapSSLConfig` attribute value on the LDAP group object If no protocol and cipher is defined using this attribute, the default configuration specified in the `ldapBindRestrictions` is followed. For more information, see "Configuring Protocols and Ciphers Using `ldapSSLConfig Attribute" on page 380`. **NOTE:** `ldapSSLConfig` attribute is available from eDirectory 9.0 SP2 onwards.
ldapGroupSSLConfig	This attribute allows you to define the TLS protocols and Ciphers in the LDAP group object. By default, this attribute is disabled. This configuration attribute follows the following order of precedence: ◆ Presence of `ldapSSLConfig` attribute value on the LDAP server object ◆ Presence of `ldapSSLConfig` attribute value on the LDAP group object If no protocol and cipher is defined using this attribute, the default configuration specified in the `ldapBindRestrictions` is followed. For more information, see "Configuring Protocols and Ciphers Using `ldapSSLConfig Attribute" on page 380`. **NOTE:** If this attribute is set through `ldapconfig get/set` command, use `ldapGroupSSLConfig` and if set through `ldif` file, use `ldapSSLConfig` with LDAP Group object DN.

Table 14-1 *Combination Values of ldapbindrestrictions and Cipher Levels*

ldapbindrestriction	Certificate	Cipher Level	Combination Value
None	RSA	Export	0
	RSA	High	48
	RSA	Medium	32
	RSA	Low	16
	ECDSA 256/384	SUITEB128	64
	ECDSA 256	SUITEB128ONLY	80
	ECDSA 384	SUITEB192	96
Disallows anonymous simple bind	RSA	Export	1
	RSA	High	49
	RSA	Medium	33
	RSA	Low	17
	ECDSA 256/384	SUITEB128	65
	ECDSA 256	SUITEB128ONLY	81
	ECDSA 384	SUITEB192	97
Disallow local bind	RSA	Export	2
	RSA	High	50
	RSA	Medium	34
	RSA	Low	18
	ECDSA 256/384	SUITEB128	66
	ECDSA 256	SUITEB128ONLY	82
	ECDSA 384	SUITEB192	98
Disallow anonymous simple bind and unbind	RSA	Export	3
	RSA	High	51
	RSA	Medium	35
	RSA	Low	19
	ECDSA 256/384	SUITEB128	67
	ECDSA 256	SUITEB128ONLY	83
	ECDSA 384	SUITEB192	99

ldapbindrestriction	Certificate	Cipher Level	Combination Value
Disallows unauthenticated bind	RSA	Export	4
	RSA	High	52
	RSA	Medium	36
	RSA	Low	20
	ECDSA 256/384	SUITEB128	68
	ECDSA 256	SUITEB128ONLY	84
	ECDSA 384	SUITEB192	100
Disallows anonymous and unauthenticated bind	RSA	Export	5
	RSA	High	53
	RSA	Medium	37
	RSA	Low	21
	ECDSA 256/384	SUITEB128	69
	ECDSA 256	SUITEB128ONLY	85
	ECDSA 384	SUITEB192	101
Disallows non-anonymous simple bind and unauthenticated bind	RSA	Export	6
	RSA	High	54
	RSA	Medium	38
	RSA	Low	22
	ECDSA 256/384	SUITEB128	70
	ECDSA 256	SUITEB128ONLY	86
	ECDSA 384	SUITEB192	102
Disallows anonymous simple bind, non-anonymous simple bind and unauthenticated bind	RSA	Export	7
	RSA	High	55
	RSA	Medium	39
	RSA	Low	23
	ECDSA 256/384	SUITEB128	71
	ECDSA 256	SUITEB128ONLY	87
	ECDSA 384	SUITEB192	103

Attributes on the LDAP Group Object

Use the LDAP Group object to set up and manage the way LDAP clients access and use the information on the OpenText LDAP server.

To require TLS for simple binds, see “Requiring TLS for Simple Binds with Passwords” on page 385. This attribute specifies whether the LDAP server allows transmission of passwords in clear text from an LDAP client. Values=0 (no) or 1 (yes).

To specify a default referral, referralIncludeFilter, referralExludeFilter and how LDAP servers process LDAP referrals, see “Using Referrals” on page 394.

To specify the TLS protocols and Ciphers, you can use the `ldapSSLConfig` attribute. For more information, see “Configuring Protocols and Ciphers Using `ldapSSLConfig` Attribute” on page 380.

Configuring Protocols and Ciphers Using `ldapSSLConfig` Attribute

OpenText eDirectory allows you to define various TLS parameters and Ciphers required for TLS communication of LDAP server. From 9.30 onwards only TLS v1.2 and TLS1v.3 protocols are supported. Any versions below TLSv1.1 protocols are not supported.

You can specify the protocol and ciphers in JSON format in the `ldapSSLConfig` attribute for both LDAP server and group object. For example, you can define the protocols and ciphers as mentioned in the below JSON format:

```
{
    "Version": 1,
    "Info": {
        "Protocol": "+ALL",
        "Ciphers": "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384", "Ciphersuites" : "TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384"
    }
}
```

NOTE: If you specify incorrect information in the `ldapSSLConfig` attribute, the default configuration specified in the `ldapBindRestrictions` will be followed.

Supported Ciphers

OpenText eDirectory CE 24.4 (v9.3) supports the following Ciphers.

TLSv1.3 supported Ciphers -

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256 [Non FIPS Only]
- TLS_AES_128_GCM_SHA256

TLSv1.2 supported Ciphers

- ECDHE-ECDSA-AES256-GCM-SHA384

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-CHACHA20-POLY1305
- ECDHE-RSA-CHACHA20-POLY1305 [Non FIPS Only]
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- AES256-GCM-SHA384
- AES128-GCM-SHA256
- AES256-SHA256
- AES128-SHA256

Configuring Ciphers

You can configure your own list of ciphers using the OpenSSL Cipher List Format. The following examples illustrate the use of Cipher list format that are used during TLS communication of LDAP server:

- For RSA certificates: `!CAMELLIA:!DH:!SRP:!MD5:HIGH+aRSA`
- For ECDSA certificates: `HIGH+aECDSA`
- For Suite B 128-bit compliant cipher suite with ECDSA certificates: `ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256`
- For Suite B 192-bit compliant cipher suite with ECDSA certificates: `ECDHE-ECDSA-AES128-GCM-SHA256`

NOTE: Run the following command to find the list of TLS cipher suites supported by the LDAP server:

```
bash$ nmap --script /usr/share/nmap/scripts/ssl-enum-ciphers.nse -p 636
<eDirectory-server-IP>
```

For example,

```
bash$ nmap --script /usr/share/nmap/scripts/ssl-enum-ciphers.nse -p 636
192.168.1.1
```

For more information on Cipher List Format, refer to the OpenSSL Ciphers (http://www.openssl.org/docs/man1.0.2/apps/ciphers.html) documentation.

Configuring Protocols

OpenText eDirectory gives you the flexibility to configure the list of protocols required during the TLS communication. To control the list of protocols, define the required protocol in JSON format in the `ldapSSLConfig` attribute. You can configure the following protocol strings:

- SSLv3, TLSv1.0 and TLSv1.1 - From OpenText eDirectory 9.30 onwards these versions of protocols are not supported.
- TLSv1.2
- TLSv1.3
- ALL

Each protocol string should be preceded by a “+” or a “-” symbol. The “+” symbol indicates that the protocol string(s) are allowed and the “-” symbol indicates that the protocol string(s) are not allowed by OpenText eDirectory. The following table lists a few TLS protocol configurations:

Protocol Configuration	Description
`+TLSv1.2`	Allows only TLSv1.2
`+TLSv1.3`	Allows only TLSv1.3
`+ALL-TLSv1.2`	Allows all except TLSv1.2
`+ALL-TLSv1.3`	Allows all except TLSv1.3
`+ALL`	Allows TLSv1.2, TLSv1.3

NOTE: A protocol can only be preceded by “-” symbol when `+ALL` is specified.

Examples:

Configuring Protocols and Ciphers in Suite B Compliant Mode for TLSv1.2

```
{
    "Version": 1,
    "Info": {
        "Protocol": "+TLSv1.2",
        "Ciphers": "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384"
    }
}
```

In the above example, protocol is defined as `+TLSv1.2` in JSON format.

Configuring Protocols and Ciphers in Suite B Compliant Mode for TLSv1.3

```
{
    "Version": 1,
    "Info": {
        "Protocol": "+TLSv1.3",
        "Ciphersuites": "TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256"
    }
}
```

In the above example, protocol is defined as +TLSv1.3 in JSON format.

Configuring Protocols and Ciphers in Non-Suite B Compliant Mode

Example-1:

```
{
    "Version": 1,
    "Info": {
        "Protocol": "+ALL",
        "Ciphers": "HIGH+aECDSA"
    }
}
```

Example-2:

```
{
    "Version": 1,
    "Info": {
        "Protocol": "+ALL",
        "Ciphers": "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384","Ciphersuites" : "TLS_AES_256_GCM_SHA384"
    }
}
```

In the above examples, protocol is defined as +ALL in JSON format which means all supported protocols are allowed during TLS communication.

Refreshing the LDAP Server

After you change a configuration option or setting on an LDAP server, you must refresh the server so that the changes can take effect.

However, you can't refresh the server while LDAP requests are being serviced. For example, if an operation requires a 15-minute walk of the OpenText eDirectory tree, the refresh won't occur until after that operation is complete.

Similarly, you can't take the LDAP server down while LDAP server threads are at work.

When a refresh is scheduled to occur, the LDAP server delays new LDAP requests from starting until after the refresh occurs.

By default, at 30-minute intervals the LDAP server checks the time stamps on the LDAP Server object and the LDAP Group object for changes to settings. If settings have changed, the server then implements the changes.

If the server discovers that time stamps on the settings have not changed, no refresh occurs. If you force a refresh, the server ignores time stamps and makes the changes.

To refresh the LDAP server, do one of the following:

- Wait for the server to reconfigure itself at the refresh interval.
- Unload and then reload `nldap.nlm`.

  You don't have to unload any prerequisite NLM programs before unloading `nldap.nlm`.

  `Nldap.nlm` unloads and then reloads dependent NLM programs.
- At the command line, change the refresh interval.

  This option might be useful if you have WAN links that are not up continuously. You can temporarily make the server's heartbeat longer or shorter, as needed.

  This change is not persistent. You must re-enter the command each time that you load `nldap.nlm`.

  At the server console, enter

  `ldap refresh [=] [`*`date`*`][`*`time`*`][`*`interval`*`]`

  - The format for the date variable is mm:dd:yyyy. If you enter zeros for all date fields, the current date is used.
  - The format for the time variable is hh:mm:ss. If you enter zeros for all time fields, the current time is used.
  - The format for the interval variable is 0 or between 1 and 2147483647 minutes. If you enter zero, the default of 30 minutes is used.

  You can add this command to the `autoexec.ncf` file in the `sys:\system` directory. Place the command after the line that loads `nldap.nlm`.

Authentication and Security

This section contains information on the following:

- “Requiring TLS for Simple Binds with Passwords” on page 385
- “Starting and Stopping TLS” on page 385
- “Configuring the Server for TLS” on page 386
- “Configuring the Client for TLS” on page 387
- “Exporting the Trusted Root” on page 388
- “Authenticating with a Client Certificate” on page 389
- “Using Certificate Authorities from Third-Party Providers” on page 389
- “Creating and Using LDAP Proxy Users” on page 389
- “Using SASL” on page 391
- “Using NMAS Based Logins for LDAP Authentication” on page 393

Requiring TLS for Simple Binds with Passwords

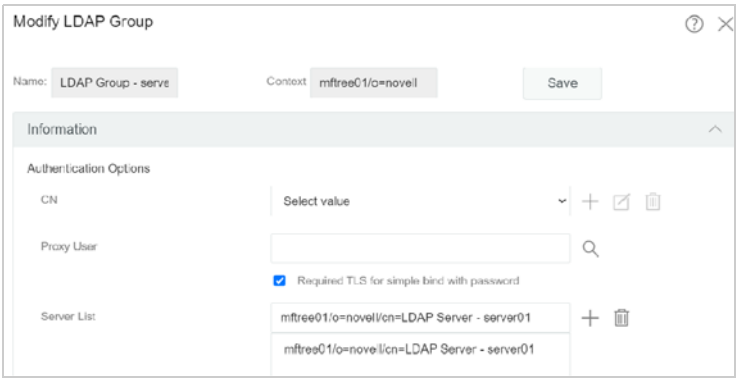
Secure Socket Layer (SSL) 3.1 was released through Netscape. IETF took ownership for that standard by implementing Transport Layer Security (TLS) 1.0. TLS 1.0 has backward compatibility with SSLv2 and v3.

TLS allows for connections to be encrypted in the Session layer. The encrypted port doesn't have to be used to get a TLS connection. There's another way: port 636 is the implied TLS port and the LDAP server automatically starts a TLS session when a client connects to the secure port.

A client can also connect to the clear-text port and later use TLS to upgrade the connection to an encrypted connection.

To require TLS for simple binds with passwords:

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click the LDAP `Group` object, then click **Information** drop down menu.
3. Check the **Required TLS for simple binds with passwords** check box.



4. Click **Save**, then click **OK**.

Starting and Stopping TLS

The extended LDAP operation STARTTLS enables you to upgrade from a clear connection to an encrypted connection. This upgrade was new to OpenText eDirectory 8.7.

When you use the encrypted connection, the entire packet is encrypted. Therefore, sniffers are unable to diagnose data sent across the network.

Scenario: Using STARTTLS— You create a clear connection (to port 389) and do some anonymous searches. However, when you get into secure data, you prefer to start a TLS session. You issue a STARTTLS extended operation to upgrade from a clear connection to an encrypted connection. Your data is secure.

You stop TLS to turn an encrypted session into a clear connection. A clear connection requires less overhead because data to and from the client is not encrypted and decrypted. Therefore, data moves faster when you use a clear connection. At this point, the connection is downgraded to Anonymous.

When you authenticate, you use the LDAP Bind operation. Bind establishes your ID based on your provided credentials. When you stop TLS, the LDAP service removes any authentication previously established. Your authentication state changes to Anonymous. Therefore, if you want a state other than Anonymous you must reauthenticate.

Scenario: Reauthenticating— Henri runs STOPTLS. His status changes to Anonymous. To access and use his files on the Net, Henri runs the Bind command, provides his login credentials, is authenticated, and continues working in clear text on the Internet.

Configuring the Server for TLS

When a TLS session is instantiated, a handshake occurs. The server and the client exchange data. The server determines how the handshake occurs. To establish that the server is legitimate, the server always sends the server's certificate to the client. This handshake guarantees to the client that the server is indeed the expected server.

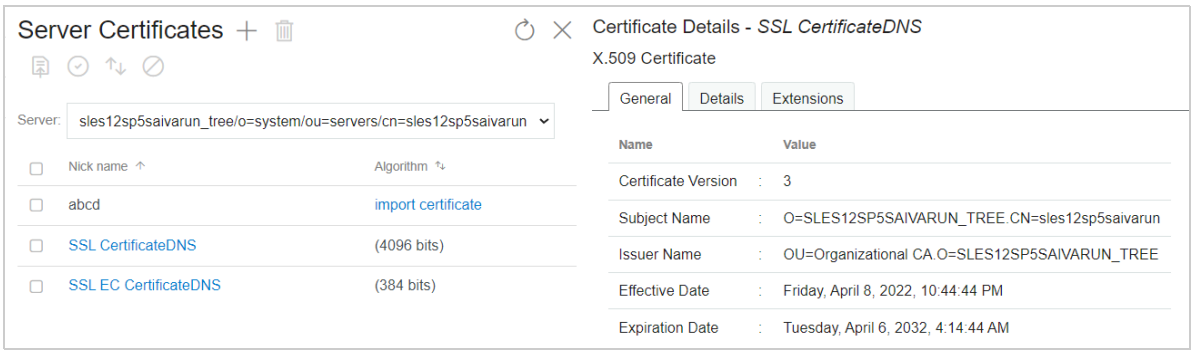
To require that the client also establish legitimacy, you set a value on the server. This attribute is ldapTLSVerifyClientCertificate.

Value	Description
0	Off. During a handshake, the server provides a certificate to the client. The server never requires the client to send a certificate. The client can use or ignore the certificate. A secure session is established.
1	During the handshake, the server provides a certificate to the client and requests a certificate from the client. The client can choose to send its certificate back. The client's certificate is validated. If the server cannot validate the client's certificate, the connection is terminated. If the client doesn’t send a certificate, the server maintains the connection.
2	During the handshake, the server requests and requires a certificate from the client. If the client does not provide a certificate, or if the certificate can't be validated, the connection is terminated.

Before the server can support TLS, you must provide the server with an X.509 certificate that the server can use to establish its legitimacy.

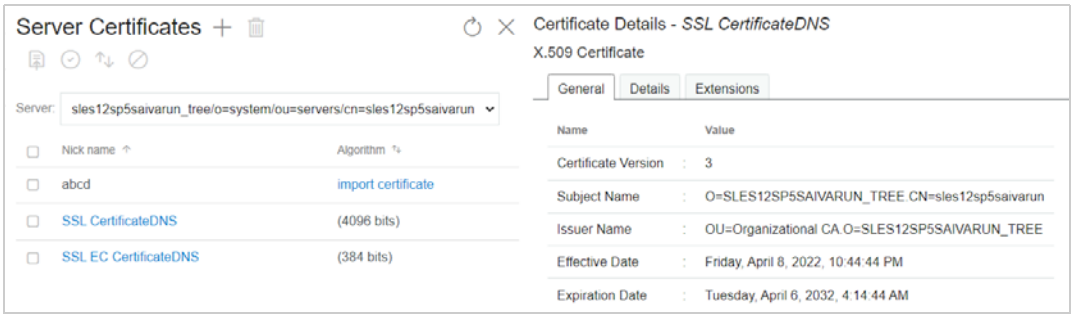
This certificate is automatically provided during the OpenText eDirectory installation. During installation, Key Material objects are created as part of Public Key Infrastructure (PKI) and NetIQ Modular Authentication Services (NMAS). The following figure illustrates these objects in OpenText Identity Console:

Figure 14-1 *Objects in OpenText Identity Console*



The installation automatically associates one of those certificates with the LDAP server. In OpenText Identity Console, the Connections tab for the LDAP Server object displays a DN. This DN represents the X.509 certificate. The Server Certificate field in the following figure illustrates this DN.

Figure 14-2 *Server Certificate Field*



In OpenText Identity Console, you can browse to the Key Material object (KMO) certificates. Using the drop-down list, you can change to a different certificate. Either the DNS or the IP certificate will work.

As part of the validation, the server should validate the name (the hard IP address or the DN) that is in the certificate.

To establish a TLS connection, ensure the following:

- The LDAP server must know the server's KMO
- You connect to the secure port or start TLS after connecting to the clear port

After you reconfigure the LDAP server, refresh the server. See “Refreshing the LDAP Server” on page 383.

Configuring the Client for TLS

An LDAP client is an application (for example, Internet Explorer or ICE). The client must understand the certificate authority that LDAP server uses.

IMPORTANT: OpenText eDirectory 9.1 onwards, all LDAP utilities including `ndsindex` and `ice` will accept certificates in `.PEM` format only. For more information on how to use the `.PEM` certificates in LDAP operations, see "Using LDAP Tools on Linux" on page 349.

When an OpenText eDirectory tree is configured, by default the configuration creates:

- A certificate authority for the tree (the tree CA).
- A KMO from the tree CA.

The LDAP server uses this certificate provider.

The client needs to import a certificate that the client will trust so that the client can validate the tree CA that the LDAP server claims to be using. The client must import a certificate from the server so that whenever the server sends its certificate, the client can validate it and verify that the server is who it claims to be.

So that the client can get a secure connection, the client must be configured before the connection.

The way that the client imports the certificate differs, based on the kind of application being used. Each application must have a method to import a certificate. IE has one way, and ICE has another way. These are different LDAP clients. Each client has its method for locating the certificates that it trusts.

Exporting the Trusted Root

You can automatically export the trusted root while accepting the certificate server.

To manually export the trusted root, see "Exporting a Trusted Root or Public Key Certificate" on page 663.

The Export functionality will create the specified file. Although you can modify the filename, it's a good idea to leave "DNS" or "IP" in the filename, so that you can recognize the type of material object. Also leave the servername.

Install the self-assigned CA in all browsers that establish secure LDAP connections to OpenText eDirectory.

If you are using the certificate with Microsoft products (for example, Internet Explorer), leave the `.der` extension.

If applications or SDKs require the certificate, import it into a certificate database.

Internet Explorer 5 exports root certificates automatically with a registry update. The traditional .X509 extension used by Microsoft is required.

Authenticating with a Client Certificate

Mutual Authentication requires a TLS session and a client certificate. Both the server and the client must verify that they are the objects that they claim to be. The client certificate was validated at the Transport layer. However, at the LDAP protocol layer, the client is anonymous until the client issues an LDAP bind request.

Up to this point, the client has proven its authenticity to the server but not to LDAP. If a client wants to authenticate as the identity contained in the client certificate, the client binds by using the SASL EXTERNAL mechanism.

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **LDAP Servers**, then click the name of an LDAP server object.
3. Click **Information** drop down menu.
4. Click the **Client Certificate** drop-down, then select **Required**.

   This enables Mutual Authentication.
5. Click **Save**, then click **OK**.

Using Certificate Authorities from Third-Party Providers

During the OpenText eDirectory installation, the LDAP server receives a tree Certificate Authority (CA). The LDAP Key Material object is based on that CA. Any certificate that a client sends to the LDAP server must be able to be validated through that tree CA.

LDAP Services for OpenText eDirectory supports multiple certificate authorities. OpenText's tree CA is just one certificate authority. The LDAP server might have other CAs (for example, from VeriSign*, an external company.) This additional CA is also a trusted root.

To configure the LDAP server to use multiple certificate authorities, set the ldapTLSTrustedRootContainer attribute on the LDAP server object. By referencing multiple certificate authorities, the LDAP server allows a client to use a certificate from an external authority.

Creating and Using LDAP Proxy Users

OpenText eDirectory assigns a [Public] identity to users who are not authenticated. In the LDAP protocol, an unauthenticated user is an Anonymous user. By default, the LDAP server grants Anonymous users the rights of the [Public] identity. These rights enable unauthenticated OpenText eDirectory and Anonymous LDAP users to browse OpenText eDirectory by using [Public] rights.

The LDAP server also allows Anonymous users to use the rights of a different proxy user. That value is located on the LDAP Group object. In OpenText Identity Console, the value is named the Proxy User field. The following figure illustrates this field in OpenText Identity Console:

Figure 14-3 *LDAP Groups*



The proxy user is a Distinguished Name. You can grant that proxy identity different rights than the Public identity has. With the proxy user, you can control LDAP Anonymous access to specific containers in the OpenText eDirectory tree.

NOTE: Don't set login restrictions for the proxy user unless you want to have them apply to all Anonymous LDAP users.

Scenario: Setting Up an NLDAP Proxy User— Digital Airlines has contracted with DataSure, a research group. DataSure will use LDAP to access and store research on DigitalAir43, a Linux server at Digital Airlines. You don't want DataSure to have Public rights to directories on DigitalAir43.

Therefore, you create an LDAP proxy user and assign that user specific rights to the DataSure directory. You populate the proxy Distinguished Name on the LDAP Group object and refresh the server. The server automatically starts using the proxy user rights for any new or existing Anonymous users.

1. On OpenText Identity Console home page > click **User Management** tile > + **Add User**, then create a proxy user without password.
2. Assign a null password to that user.
3. (Optional) Assign the proxy user rights to specified directories.
4. On Identity Console home page click **LDAP Configuration** > form **Type** drop down select **LDAP Group** > click **Search** > select the LDAP `Group` object.
5. Click **Information**. In the **Proxy User** field, click 🔍 to browse and select the LDAP`Proxy` user that has been already created, then click **Save**.
6. Click **OK**.

Using SASL

Simple Authentication and Security Layer (SASL) is a mechanism for adding authentication support and data security services to connection-based protocols through different mechanisms. It presents a well-formed interface between the protocols and mechanisms. In addition, it provides a protocol for securing subsequent protocol exchanges within a data security layer along with data integrity, data confidentiality, and other services.

SASL is designed to allow new protocols to reuse the existing mechanisms without requiring redesign of the mechanisms, and it also allows existing protocols to make use of new mechanisms without the redesign of protocols. To use SASL, each protocol provides a method for identifying which mechanism is to be used, a method for exchange of mechanism-specific server-challenges and client-responses, and a method for communicating the outcome of the authentication exchange.

SASL mechanisms are named by strings, consisting of uppercase letters, digits, hyphens, and underscores. SASL mechanism names must be registered with the Internet Assigned Numbers Authority (IANA).

If a server supports the requested mechanism, it initiates an authentication protocol exchange. This consists of a series of server challenges and client responses that are specific to the requested mechanism. During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity from the client to server, and negotiates the use of a mechanism-specific security layer. If the use of a security layer is agreed upon, then the mechanism must also define or negotiate the maximum cipher-text buffer size that each side is able to receive.

The LDAP server supports the following mechanisms:

- DIGEST-MD5
- EXTERNAL
- NMAS_LOGIN
- GSSAPI

These mechanisms are installed on the server during an OpenText eDirectory installation or upgrade. However, on Linux, the nmasinst utility must be used to install the NMAS methods.

As specified above, the LDAP server queries SASL for the installed mechanisms when it gets its configuration, and automatically supports whatever is installed. The LDAP server also reports the current supported SASL mechanisms in its rootDSE by using the supportedSASLMechanisms attribute. Because these are the registered mechanisms, the correct naming conventions must be used to make use of them.

The LDAP bind protocol allows the client to use various SASL mechanisms for authentication. When the application uses the LDAP bind API, it must choose either the simple bind and supply a DN and password, or choose the SASL bind and supply the SASL mechanism name and the associated SASL credentials required by the mechanism.

DIGEST-MD5

LDAP supports the DIGEST-MD5 mechanism through the bind request. Instead of requesting an LDAP simple bind (DN and clear-text password), you request an LDAP SASL bind by providing the DN and the MD5 credentials. The DIGEST-MD5 mechanism does not require TLS. The LDAP server supports DIGEST-MD5 over clear and secure connections.

MD5 provides an encrypted hash of passwords. Passwords are encrypted even on clear connections. Therefore, the LDAP server accepts passwords that use MD5 on either the clear-text or encrypted port. If someone tries to sniff this connection, the password cannot be detected. However, the entire connection can be spoofed or hijacked.

This mechanism is an LDAP SASL bind (not a simple bind). Therefore, the LDAP server accepts these requests, even if you selected the **Require TLS for Simple Binds with Passwords** check box during installation.

EXTERNAL

The EXTERNAL mechanism informs the LDAP server that the user DN and credentials have already been supplied to the server. Therefore, the DN and credentials do not need to come across in the bind request.

The LDAP bind request uses the SASL EXTERNAL mechanism to instruct the server to do the following:

- Ask an EXTERNAL layer what the credentials were
- Authenticate the user with those credentials and user

After this is done, a secure handshake occurs. The LDAP server requests credentials from the client and the client passes them to the server, then the server receives the certificate that was passed from the client, passes the certificate to the NMAS module, and authenticates the user as whatever DN was supplied in the certificate

Having a certificate with a usable DN requires some setup on the client. For information about setting up the certificate, see the NMAS online documentation (https://www.netiq.com/documentation/edir88/nmas88/data/bookinfo.html).

Even if the client sends an EXTERNAL mechanism, the LDAP server could fail the request.The following could be possible reasons for failure:

- The connection is not secure.
- Although the connection is secure, the client did not provide the required certificate during the handshake.
- The SASL module is unavailable.

NMAS_LOGIN

NMAS is a development framework that allows you to write applications that authenticate to the network using various login and authentication methods. The NMAS framework allows you to design a flexible and expandable login and authentication system using modular plug-in methods that leverage NICI and OpenText Directory Services (OpenText eDirectory).

The NMAS_LOGIN mechanism provides the LDAP server with the biometrics capability of NMAS. For more information, see the NetIQ Modular Authentication Services NDK (http://www.novell.com/documentation/developer/nmas/).

GSSAPI

The GSSAPI mechanism enables a Kerberos user to authenticate to an OpenText eDirectory server using a ticket, without needing to enter a separate LDAP user password. This functionality is targeted at LDAP application users in environments that already have the Kerberos infrastructure in place. Such users must be able to use the Kerberos server-issued tickets to authenticate to the LDAP server without providing a separate LDAP user password.

For information on configuring GSSAPI, refer to Appendix E, “Configuring GSSAPI with OpenText eDirectory,” on page 783.

Using NMAS Based Logins for LDAP Authentication

The NMAS login is enabled by default in OpenText eDirectory. To disable the NMAS login, set `NDSD_TRY_NMASLOGIN_FIRST` to `false`.

NOTE: You must add all the environment variables required for the OpenText eDirectory service to run in the pre_ndsd_start_custom script on RHEL 7.x and SLES 12.x platforms.

Using the LDAP Server to Search the Directory

This section contains information on the following:

- “Setting Search Limits” on page 393
- “Using Referrals” on page 394

Setting Search Limits

The following attributes on the LDAP server object control how the LDAP server searches the Directory:

- Search Entry Limit

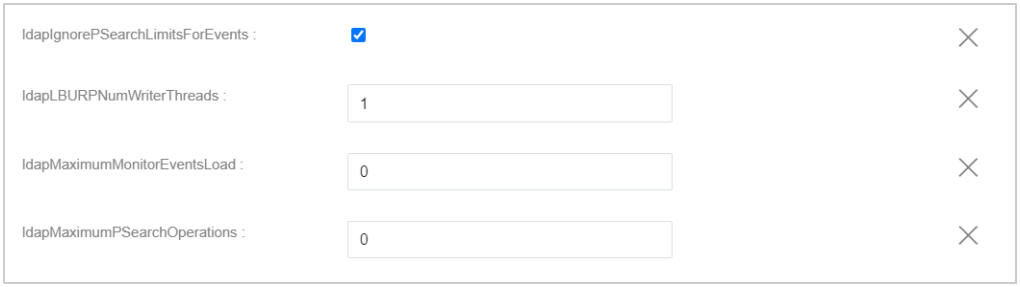
  Limits the size of a search. The default is 0, for no limit on size. So that the LDAP server isn't overloaded, you can limit the number of entries that the LDAP server returns from a search request.

Scenario: Limiting the Size of a Search— Henri requests a search that could result in thousands of replies concerning objects that the search finds. However, you have set a limit of 10 results. LDAP server stops searching after returning 10 results. A system message informs Henri that the search has ended even though more data is available.

- Search Time Limit

  Limits the time that the server searches. The default is 0 seconds, for no time limit.

The following figure illustrates these attributes in OpenText Identity Console.



1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **LDAP Servers**.
3. Click **Others** drop down menu.
4. Scroll to the Restrictions section, enter values, then click **Save**.

The client can also set limit search requests (for example, limiting the search to two seconds). If the client limit conflicts with the server limit, the LDAP server uses the lowest or smallest value from either request.

The search is based on Access Control Lists (ACLs). Therefore, an Anonymous search could yield the few entries that Public is allowed to view, even though thousands of entries exist in the Directory.

Using Referrals

A referral is a client-centric method to resolve names. An LDAP client sends a request to an LDAP server, which attempts to find the target entry of the operation locally. If the server can't find the target entry, the server uses the knowledge references that it has to generate a referral to a second LDAP server that knows more about the entry. The first server sends the referral information to the LDAP client.

The LDAP client then establishes a connection to the second LDAP server and retries the operation. If the second LDAP server has the target entry of the operation, it performs the operation. Otherwise, the second server also sends a referral back to the client. This process continues until one of the following occurs:

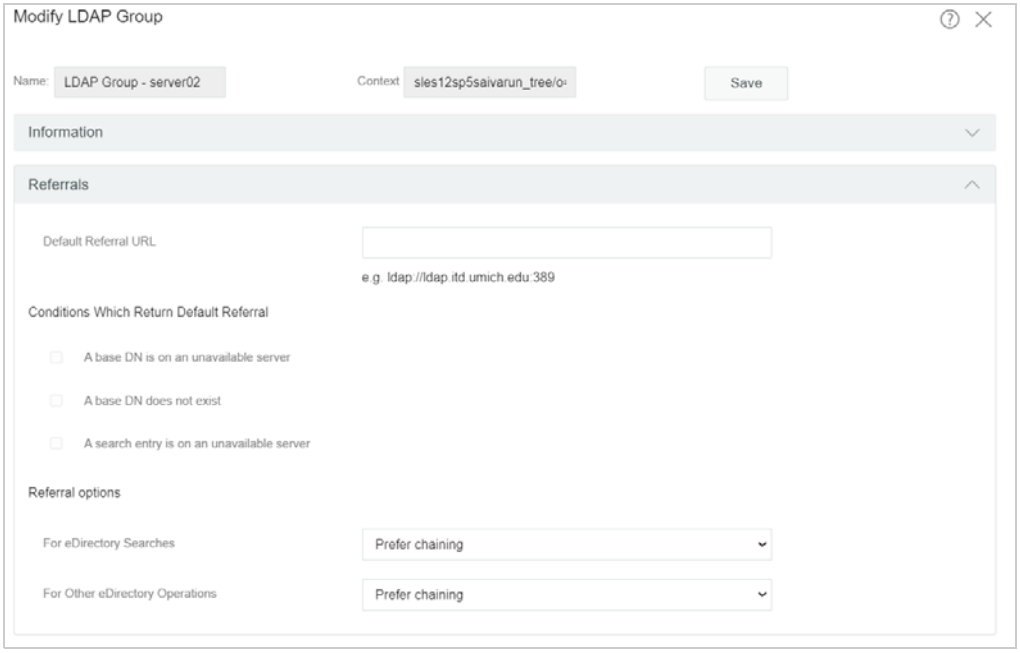
- The client contacts a server that has the entry and can perform the desired operation
- The LDAP server returns an error indicating that the entry doesn’t exist
- The LDAP server indicates that no more referrals can be followed

A functionality introduced in LDAP for OpenText eDirectory 8.7 causes referrals to behave slightly differently than with earlier versions of OpenText eDirectory and NDS. The differences influence the way you configure LDAP Services.

Default Referrals

Typically, a default referral URL contains an LDAP URL that points to a server that holds the root of the tree. An LDAP URL has the following form: `ldap://host:port`.

You enter a default referral in the Default Referral URL field:



Historically, the OpenText eDirectory LDAP server sent the default referral in a number of failover situations. Many users find these behaviors strange and sometimes unpredictable. LDAP Services for OpenText eDirectory lets you control when the default referral is sent for any kind of subordinate referral.

The new option is a value (setting) held on the ldapDefaultReferralBehavior attribute on the LDAP server and LDAP Group objects. The value is an integer which is a bitmask of the following bits.

Bits	Value
0x00000001	The base DN is not found
0x00000002	The base DN is on an unavailable OpenText eDirectory server
0x00000004	An entry in the search scope is on an unavailable OpenText eDirectory server

If the LDAP server is configured to Always Refer for the operation, and if any of the conditions listed are met and the corresponding value is set, the default referral is returned.

Setting Referrals for Search Operations

A functionality introduced in LDAP for eDirectory 8.7 causes referrals to behave slightly differently than with earlier versions of OpenText eDirectory and NDS. The differences influence the way you configure LDAP Services.

You can configure the eDirectory LDAP server to return referrals to other OpenText eDirectory servers within the OpenText eDirectory tree. By default, the LDAP server chains all operations to other OpenText eDirectory servers on behalf of the user, and referrals are never returned.

Prior to eDirectory 8.7, the referral options only existed as settings on the LDAP Group object. With OpenText eDirectory 9.0 or later, you can set these options on the LDAP server object also. Any setting on the LDAP server object overrides that setting on the LDAP Group object.

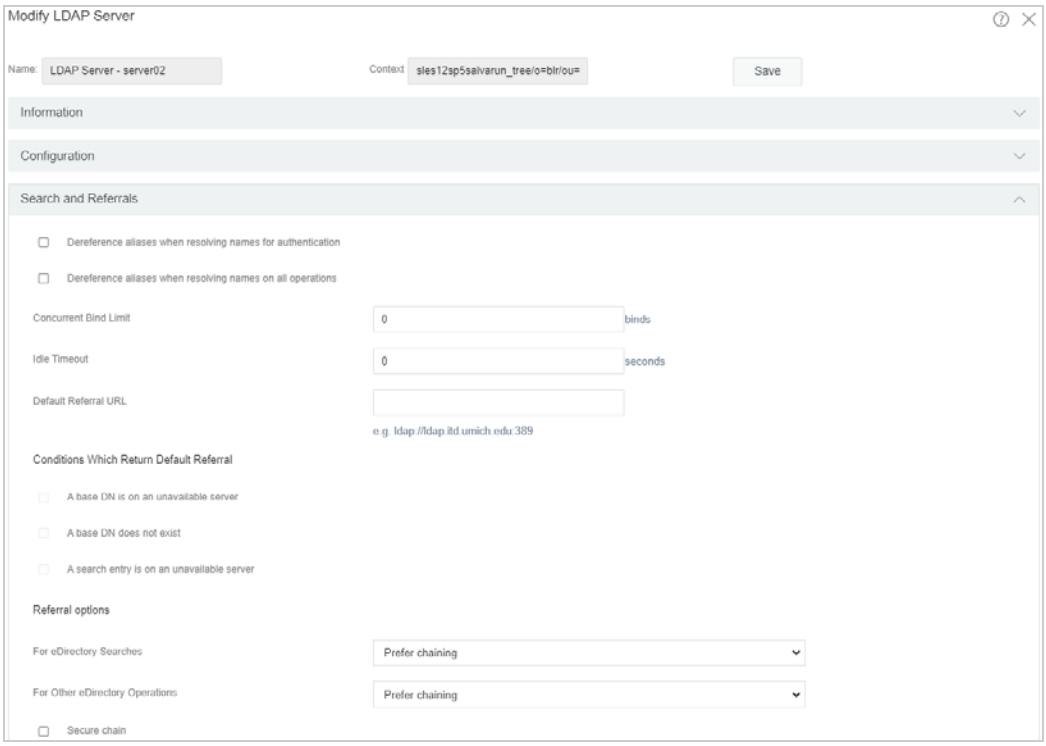
You set the Referral Option by manipulating the ldapSearchReferralOption attribute. Previous to LDAP Services for eDirectory 8.7, you could set this attribute to the following options:

- “Prefer Chaining” on page 398 (the default option)
- “Prefer Referrals” on page 399
- “Always Refer” on page 399

These referral options apply only to referring and chaining to other OpenText eDirectory servers within the OpenText eDirectory tree. These configuration settings don’t control referrals that come from a nonauthoritative partition. Therefore, even though you select an option (for example, Always Chain) from the Referral Options drop-down list, referrals will still come from nonauthoritative partitions to other servers.

To support superior referrals to OpenText non-eDirectory DSAs, LDAP Services for eDirectory 8.7.a has an Always Chain option. See “Always Chain” on page 397.

The following figure illustrates the LDAP referral drop-down lists for searches and other operations.



“Other” OpenText eDirectory operations include referrals for the Add, Delete, Modify, and Bind operations.

Always Chain

The Always Chain option is a “never refer” option. If you select this option, the OpenText eDirectory LDAP server never returns referrals to other OpenText eDirectory servers in the OpenText eDirectory tree. The LDAP server checks with other LDAP servers on behalf of the requesting client and sends the referral to the client.

The Always Chain option will be most beneficial to you if you have an OpenText eDirectory deployment that participates as subordinate servers in a global federated tree.

These referral options only apply to the way referrals are handled within the eDirectory tree. They have no bearing on referral behavior to OpenText non-eDirectory servers.

The reason for blocking referrals to other OpenText eDirectory servers is subtle, but may prove invaluable. If the nonauthoritative data on an eDirectory 8.7 or later server is replicated to another, older OpenText eDirectory server, a referral to the older server might cause a client application to get a distorted view of the global tree.

For example, assume that an LDAP client caches referrals to LDAP servers and sends requests to the server it last communicated with. If the client is configured to send requests to an OpenText eDirectory server that supports superior referrals, the client's view of the global tree should be normal.

However, LDAP servers earlier than eDirectory 8.7 don't understand nonauthoritative areas and superior referrals. Therefore, if the client follows a referral to an earlier-version OpenText eDirectory server in the OpenText eDirectory tree, and continues to send requests to that earlier-version server, the earlier-version LDAP server will present the nonauthoritative data as if it were the actual directory tree data.

An intelligent client should, however, interrogate the supportedFeatures attribute of the rootDSE to ascertain whether or not the server supports superior referrals.

Prefer Chaining

The Prefer Chaining option indicates that search operations will not normally return referrals. Instead, the LDAP server progresses the search operation across all OpenText eDirectory DSAs required to complete it.

The exception is a search operation that is accompanied by the persistent search control. In this case, because the OpenText implementation of persistent search does not support chaining, referrals are sent if the scope of the search operation is not all held locally.

The LDAP server receives a search operation. If the entry in the tree is not stored locally, the server automatically chains to other servers. After the entry has been located, the LDAP server acts as proxy for the LDAP client. Using the same identify that the LDAP client is bound with, the LDAP server authenticates to the remote server and continues the search operation there.

The LDAP server that received the original search request sends the LDAP client all search entries and the search result. Because the LDAP server fully takes care of the request, the LDAP client is unaware that other servers were involved.

Through chaining on OpenText eDirectory, an LDAP server that doesn't have much data can appear to hold the data of the entire tree.

Prefer Chaining is important concerning partitions.

Scenario: Finding Information in another Partition— At the Digital Airlines Company, Luc selects the Prefer Chaining option for LDAP server DAir43. DAir43 is in Partition A. Partition B is a subpartition of A and contains LDAP server DAir44.

An LDAP client requests a search. DAir43 searches locally for the entry but only finds part of the data. DAir43 automatically chains to DigitalAir44, which has the needed entry. DAir44 sends the data to DAir43, and DAir43 sends the entry to the LDAP client.

The Prefer Chaining option causes the LDAP server to chain to other servers for search requests (when needed) unless the operation is a Persistent Search. For information on Persistent Search, see "Persistent Search: Configuring for OpenText eDirectory Events" on page 407.

Prefer Referrals

The Prefer Referrals options indicates that search operations will return referrals to other eDirectory servers in the OpenText eDirectory tree when needed. Referrals are sent only if the local server can ensure that the server holding the data is operational and that the LDAP service is running. Otherwise, the operation is chained to the other server, or the operation fails if the other server in inoperable.

You have two partitions and are doing a subtree search. You get down to a point where the search entries are no longer held on the local server. Therefore, the search must go to another server. If the server that holds the replica of that data (that partition) is also running `nldap.nlm`, the LDAP server builds an LDAP referral and sends it back to the LDAP client.

If the server holding the replica isn't running `nldap.nlm`, LDAP server chains the request to the other server, thereby completing the search.

When `nldap.nlm` starts up, the LDAP server communicates to OpenText eDirectory that the LDAP server is a referral point. If a client has received referrals but the referrals stop, the LDAP server is not running.

Always Refer

The Always Refer option follows the same logic as Prefer Referrals, except that the Default Referral is sent under various failover situations (for example, an object is not found or the server is down).

If another server that holds the rest of the data isn't running the LDAP service, the first LDAP server won't chain the request to the second server.

If you mark the Always Refer option, you are allowed to enter a default referral. The Default Referral field enables you to glue two different vendor LDAP servers together and build your own Directory tree.

Scenario: Using a Default Server— You have an LDAP tree. One part of the tree is serviced by OpenText eDirectory. A subordinate partition is serviced by iPlanet. In the Default Referral field, you place a URL that references the iPlanet server. An LDAP client requests a search.

Unable to resolve the base DN, the LDAP server sends the client the string in the Default Referral field. The referral instructs the LDAP client to look in the place specified in the URL The LDAP client contacts the iPlanet server, which completes the search.

Whenever a default referral is configured and the server doesn’t find the base DN being searched for, the client receives the default referral.

The format for a referral is an LDAP URL. For example, LDAP://123.23.45.6:389.

When the LDAP server sends a default referral to a client (because the base DN was unavailable), the server appends an additional forward slash (/) and the DN that the client was looking for. The default referral and the appended information go to the client. The client sends the search request to the server specified in the default referral.

The LDAP Group object has a string field for the default referral. The LDAP server treats that data as a string. There is no validation. Whatever is entered is prepended to the referral. Some data is appended to the referral. The LDAP server expects the string to look like a URL.

When clients get referrals to other OpenText eDirectory servers that are running LDAP, the client receives two referrals per server:

- A referral directing the client to the clear-text port
- A referral directing the client to the secure port

To differentiate between the two referrals, the clear-text referral states `ldap://` and the secure port displays `ldaps://`.

A referral from the server appends the port number.

Setting Referrals for Other Operations

The historical referral option setting only applied to the search operation. To provide a comparable option for other operations, the ldapOtherReferralOption attribute is used. This attribute allows the same values and controls the behavior for non-search operations (excluding bind, which never sends a referral).

Referral Filtering

If you have multiple replica servers running in a tree and have configured LDAP server(s) to return referrals using the Prefer Referrals/Always Refer option, then the LDAP server will return referrals if the object identified by DN in the requested operation is not present locally. In such a case, LDAP client sends a request to the server, and the server returns a referral list of all the LDAP servers holding that object. Using this referral list, LDAP clients will follow any of these referrals to perform the operation. If the client chooses to follow the referral to a resource-starved server or a server that is located across a slow link, clients would see a slow response from the server. This in turn affects the performance of the LDAP client. Since LDAP application developers will not have complete knowledge about the servers and network configurations, the solution for this problem is to provide a referral filtering mechanism at the LDAP server to return the referrals of specific server(s). Administrators would have the requisite knowledge, e.g. the nature of LDAP servers in the network and network link speeds to make appropriate configuration of referral filtering.

Set up the referral filter on the LDAP Group object using the attributes “referralIncludeFilter” and “referralExcludeFilter”. Setting these filters in these attributes will be applicable to all the LDAP servers belonging to this LDAP Group object. The LDAP server will return all the LDAP referrals matching with the referralIncludeList filter and drop the ones that match the referralExcludeFilter filter.

If only referralIncludeFilter is specified, the LDAP referrals which match the referralIncludeFilter values will be returned to the LDAP clients and all other referrals will be excluded from the referral list. Similarly, if only referralExcludeFilter is specified, the LDAP referrals which do not match the referralExcludeFilter values will be returned to the LDAP clients. If both filters exist and the referral does not match any of these filters, it will be excluded.

If all available referrals are disallowed by the filter, the server will behave as if no referrals are available and return LDAP_OTHER (80), which some client tools report as “Unknown error.” After adding or modifying these filter attributes, if the LDAP server is not refreshed, changes will take place after the subsequent automatic refresh.

Currently, adding or modifying these filter attributes can be done only with the tab available in OpenText Identity Console.

Format to Specify LDAP Referral Filtering —The LDAP referral filter format is a simple IP address format:

`[ldap://] | [ldaps://] IPAdress[:port]`

Here, specifying the clear text port or TLS port will be same as pre-pending `ldap://` or `ldaps://` strings. If neither `ldap` or `ldaps` is specified, the match filter is applicable for both clear text as well as TLS referrals.

Examples:

Examples	Description
`1.2.3.4`	# matches both LDAP and LDAPS referrals on any port
`1.2.`	# matches all IP addresses of 1.2.X.Y
`1.2.3.`	# matches all IP addresses of 1.2.3.Y
`ldap://` or `ldap://*`	# matches all the clear text port LDAP referrals
`ldaps://` or `ldap://*`	# matches all the ssl port LDAP referrals
*	# matches all
`ldaps://5.6.7.8:636`	# matches for SSL port 636 on IP addresses 5.6.7.8

These filter attributes (`referralIncludeFilter` and `referralExcludeFilter`) are multi-valued. You can choose as many matching filters as you need.

Example Scenarios

- To make an LDAP server return only referrals with the IP address 1.2.X.Y where X = {0 to 255} and Y = {0 to 255} and exclude all others, enter the following:

  referralIncludeFilter = { 1.2 }

- To make an LDAP server return referral, that exclude all the referrals that match IP address 164.99.X.Y, where X is not equal to 100 and match 164.99.100.Y, enter the following:

  ```
  referralIncludeFilter = { 164.99.100., "*"}
  referralExcludeFilter = { 164.99. }
  ```

  Here, even though the IP address 164.99.100.Y matches `referralExcludeFilter`, since these IP addresses have more matched fields, these referrals will be returned to the LDAP clients.

  NOTE: While specifying a partial IP address, the trailing "." can be omitted.

- To make an LDAP server return only clear text port referrals and drop SSL port referrals, enter the following:

  ```
  referralIncludeFilter = { "ldap://" }
  ```

  OR

  ```
  referralExcludeFilter = { "ldaps://" }
  ```

- To make an LDAP server return from a set of IP addresses and drop all other IP address referrals, enter the following:

  ```
  referralIncludeFilter = { 1.2.3.4, 2.3.4.5:389, 3.4.5.6:636, ldaps://
  4.5.6.7 }
  referralExcludeFilter = { "*" }
  ```

  NOTE: Here, `referralExcludeFilter` is not required. Any populated `referralIncludeFilter` implies to exclude all others.

- There are two filters, as follows:

  ```
  referralIncludeFilter = { 1.2.3.4 }
  referralExcludeFilter = { 2.3.4.5 }
  ```

  A referral with IP address 3.4.5.6 will be excluded as it does not match the referralInclude filter, even though it does not match the `referralExcludeFilter` as well.

Invalid Filters —The following filters are not supported.

".2.3.4" or "*.2.3.4" will not match the IP addresses *X*.2.3.4.

"2.3.4*" will not match the IP addresses like 2.3.41 or 2.3.42.

DNS names like sever1.mydomain.com, or *.mydomain.com are not supported. Adding the port ranges to the filters like allow referral IP address on the port start-to-end is not supported.There are no validation checks done before adding these filter values to these attributes. But in case of an invalid filter, the LDAP server will ignore those filters and log the information into `ndsd.log` file.

Known Issues —The LDAP rootDSE search returns altServers if there are any replica servers in the LDAP URL format. These URLs do not get filtered using this mechanism.

No Support for ManageDsaIT

In LDAP Services for OpenText eDirectory, the distributed relationships between OpenText eDirectory servers in an OpenText eDirectory tree are managed by means other than the use of the ManageDsaIT control. The ManageDsaIT control won't allow the LDAP client to interrogate or update OpenText eDirectory subordinate or cross references.

Functionality Not Supported

LDAP Services for OpenText eDirectory doesn't support subordinate references. You cannot reliably create a nonauthoritative partition that is subordinate to an authoritative partition and have it send referrals. If you elect to do this, referrals are only sent when resolving the base DN for an operation. SearchResultReferences are not sent.

There is no support for distributed updates of data in the nonauthoritative area. If a name change occurs on the root server, there is no built-in mechanism to copy that name change to the OpenText eDirectory server holding that same data in a nonauthoritative area.

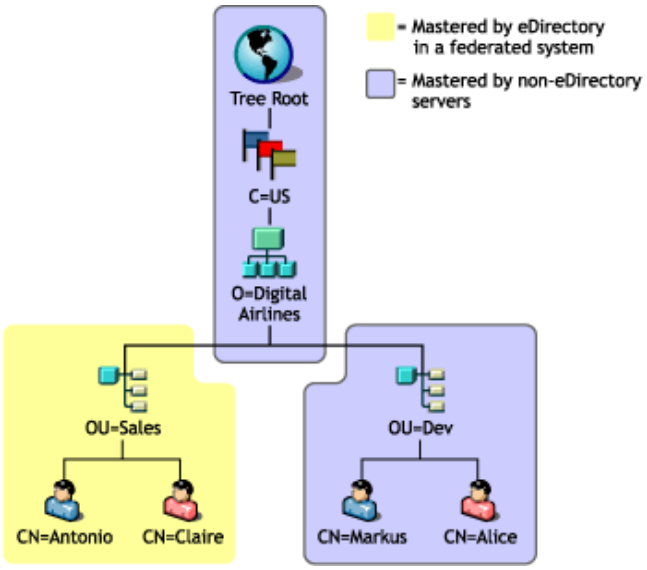
Configuring for Superior Referrals

Often, larger deployments need a directory tree that uses LDAP server software from different vendors. Such a tree is a global federated tree. LDAP Services for OpenText eDirectory has the capability to return referrals to a superior DSA in a federated tree.

Scenario: Superior Referrals in a Federated Tree

Luc is responsible for networks at Digital Airlines. An OpenLDAP server is being used to master the root of a directory tree at Digital Airlines (from the tree root down to O=Digital Airlines). An organization (OU=Sales) is mastered by an OpenText eDirectory server, and another organization (OU=Dev) is held on an iPlanet server.

The following figure illustrates this tree:



OpenText eDirectory masters only the data within the partition for OU=Sales. The data in the other areas are mastered on non-eDirectory DSAs. Luc configures LDAP Services to return superior referrals whenever an operation is rooted at O=Digital Airlines or above, or anywhere under O=Digital Airlines that is not part of the OU=Sales hierarchy.

An operation is sent to the OpenText eDirectory LDAP server with a base DN of OU=Dev,O=Digital Airlines,C=US. A referral is returned pointing to the servers holding that entry or to servers that have knowledge of the servers holding that entry.

Likewise, a subtree search rooted at O=Digital Airlines,C=US results in a referral to the root DSA. The root DSA in turn returns referrals to the DSAs mastering OU=Sales and OU=Dev.

So that the OpenText eDirectory server can participate in this tree, LDAP Services allows OpenText eDirectory to hold the hierarchical data above it in a partition marked “nonauthoritative.” The objects in the nonauthoritative area consist only of those entries needed to build the correct DN hierarchy. These entries are analogous to X.500 “Glue” entries.

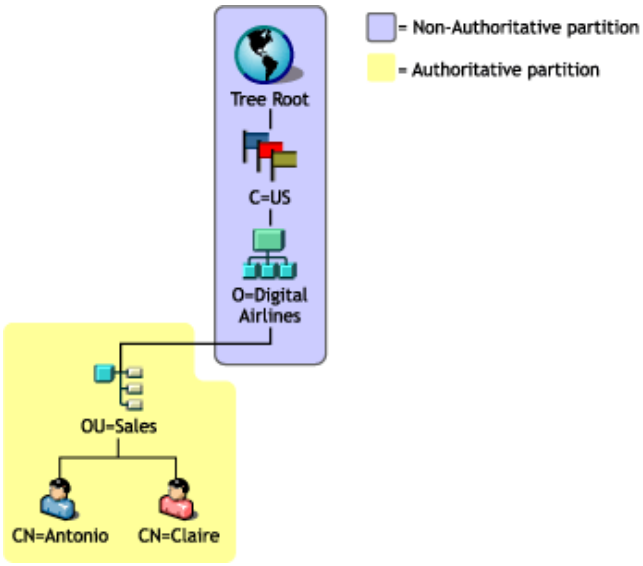
In this scenario, the Root, C=US, and O=Digital Airlines objects are held on the OpenText eDirectory server in a nonauthoritative area.

OpenText eDirectory allows knowledge information (referral data) to be placed within nonauthoritative areas.This information is used to return referrals to the LDAP client.

When an LDAP operation takes place in a nonauthoritative area of the OpenText eDirectory tree, the LDAP server locates the correct reference data and returns a referral to the client.

Creating a Nonauthoritative Area

The following figure illustrates the actual data held on the OpenText eDirectory server in the federated tree shown in “Scenario: Superior Referrals in a Federated Tree” on page 403.



Notice that entries are placed above OU=Sales, even though these entries are mastered by another DSA. This placement is necessary to provide the proper DNs for the entries mastered by the OpenText eDirectory server.

To create a nonauthoritative area:

1. Segregate the nonauthoritative data from the authoritative data.

   Create a partition boundary at the top of the authoritative area. An OpenText eDirectory server considers itself authoritative for all data that it holds unless otherwise specified.
2. Mark the root partition as nonauthoritative.
   - **2a** Add the authoritative attribute to the rootmost entry in the partition.
   - **2b** Populate the authoritative attribute with a value of zero.
3. Draw a boundary at the bottom of the nonauthoritative area.

   Create partition roots at the areas of the subtree that this server is to be authoritative for. For example, in the figure above, a partition root exists at the OU=Sales entry. The new partitions won't have the authoritative attribute set to zero. Therefore, the server will be authoritative for the partitions.
4. Refresh the LDAP server.

The LDAP server caches the authoritative and nonauthoritative area boundaries whenever its configuration is refreshed. If you don't manually refresh the server configuration, the server will automatically refresh itself on a 30-minute background task.

Multiple partitions can be stacked in a chain of nonauthoritative areas. However, LDAP Services for eDirectory requires that all nonauthoritative partitions must be contiguous and held in local replicas.

Specifying Reference Data

When the LDAP server finds that an operation is taking place in a nonauthoritative area, it looks for information it can use to return a referral to the client. This referral information might be at one of the following:

- Located on any or all of the entries in the nonauthoritative area
- Specified as a default referral on the LDAP server or LDAP Group object that holds the configuration data for the server

Referral information held on entries in the nonauthoritative area is an Immediate Superior Reference. Such referral information consists of a multi-valued ref attribute. For a description of this attribute, see RFC 3296 (http://www.ietf.org/rfc/rfc3296.txt).

Referral information held in the Default Referral configuration setting is a Superior Reference and is single-valued. See immSupr and supr DSE types in X.501.

Reference data is held in the form of an LDAP URL, but only specifies the host and (optionally) the port of the DSA being referred to. The following example illustrates this reference data:

```
ldap://ldap.digital_airlines.com:389
```

The LDAP server looks at the base DN for the operation (or if not found, the matched DN). If the base DN contains reference information, the LDAP server returns that information as a referral.

If no reference information is found, the LDAP server traverses the tree upwards, looking for reference information. If no reference information is found after exhausting all entries, the LDAP server returns the superior reference. This reference is held in the default referral setting on the LDAP Group or LDAP Server object.

Adding an Immediate Superior Reference

You can add an auxiliary object class called immediateSuperiorReference to an entry in the nonauthoritative area. This auxiliary class adds a ref attribute, which is populated with one or more LDAP URLs. Each URL points to a DSA’s host name and (optionally) port.

Adding a Superior Reference

Historically, the LDAP Group object has had an ldapReferral attribute. This attribute held a default reference that was used for various failover situations when returning referrals to other OpenText eDirectory servers in an OpenText eDirectory tree. In LDAP Services for OpenText eDirectory, this attribute is used to hold a single default referral to a superior DSA in a federated tree.

Additionally, the ldapReferral attribute has been added to the LDAP server object. If the ldapReferral attribute contains a value on the LDAP server object, that setting overrides the value held in the same attribute on the LDAP Group object. This behavior allows you to configure all LDAP servers participating in a group to have a particular default referral, while one or two servers override that value with a different default referral.

The value on the ldapReferral attribute is an LDAP URL. The URL holds the host and optional port of the DSA being referred to.

Updating Reference Information through LDAP

If you followed the steps above, in order, and used LDAP to perform the tasks, you were likely unable to add an immediate superior reference. This is because the root partition had already been marked nonauthoritative, so LDAP sends referrals for any operation acting on data within that partition.

To update or interrogate information in a nonauthoritative area, the ManageDsaIT control must accompany the LDAP request. For information on this control, see RFC 3296 (http://www.ietf.org/rfc/rfc3296.txt). This control effectively causes the LDAP server to treat the entire nonauthoritative area as though it is authoritative.

NOTE: The superior reference feature is only available through LDAP. Other protocols (for example, NDAP) are not affected by the presence of the authoritative attribute. Therefore, the use of OpenText Identity Console to interrogate and update data in the nonauthoritative area is unhindered.

Affected Operations

Nonauthoritative areas and superior referrals affect the following LDAP operations:

- Search and Compare
- Modify and Add

  DN-syntax attribute values are not checked. Therefore, a group member attribute can contain DNs that point to entries in a nonauthoritative area.
- Delete
- Rename (moddn)
- Move (moddn)

  If the parent DN falls within a nonauthoritative area, an error affectsMultipleDSAs should be returned.
- Extended

Discovering Support for Superior References

Support for superior referrals is available only in LDAP Services for OpenText eDirectory 8.7 and later. To discover whether an OpenText eDirectory server supports this functionality, you can read the supportedFeatures attribute on the root DSE. If the supportedFeatures attribute lists the OID 2.16.840.1.113719.1.27.99.1, these features are available. Additional discovery-related changes to the root DSE object include the following:

- namingContexts

  This attribute only lists the partition roots held on the local DSA that the server is authoritative for. No nonauthoritative partition roots are listed.

- altServer

  This attribute won't list other OpenText eDirectory servers that share only nonauthoritative partitions with the local server.

- superiorReference

  This attribute advertises the superior referral for the DSA. This value is administered by updating the ldapReferral attribute on the LDAP Server or LDAP Group object.

Persistent Search: Configuring for OpenText eDirectory Events

OpenText eDirectory has an event service that enables applications to be notified of significant events that occur within the Directory. Some of these events are general events that can pertain to any Directory service. Other events are specific to OpenText eDirectory and its special features.

OpenText eDirectory events are exposed to applications through two different extensions to the LDAP protocol:

- An implementation of the Persistent Search Control

  The Persistent Search feature of OpenText eDirectory is a search operation that keeps going after the initial set of matching entries is returned. Persistent Search is an extension to the LDAP v3 search operation that moves the burden of checking for updates within a search result set from the client to the server. The Persistent Search control allows the client to perform a normal LDAP search operation (specifying the base DN, scope of search, search filter, and so on) and then, rather than having the server return a `SearchResultDone` message at the end, the operation maintains a connection so the client can be updated each time an entry in the result set changes. This allows the client to maintain a cache of the entries it is interested in, or trigger some logic whenever an update occurs.

  The article "Persistent Search: A Simple LDAP Change Notification Mechanism" (http://www.ietf.org/proceedings/01mar/I-D/ldapext-psearch-03.txt) describes this extension in further detail.

- Monitor Events (an extended LDAP operation that is specific to OpenText eDirectory)

Applications that use OpenText eDirectory event services can place a heavy computational load on the directory. Various administrative parameters are available to help control how event services are used on individual OpenText eDirectory servers. These parameters are stored on the LDAP Server object. Use OpenText Identity Console to set these parameters.

Specific applications that use the event service might require that you set these parameters to specific values. The documentation for such applications will indicate specific requirements for the application.

For more information, see "Understanding and Using Persistent Search in eDirectory" (http://support.novell.com/techcenter/articles/dnd20030204.html).

Managing Persistent Searches

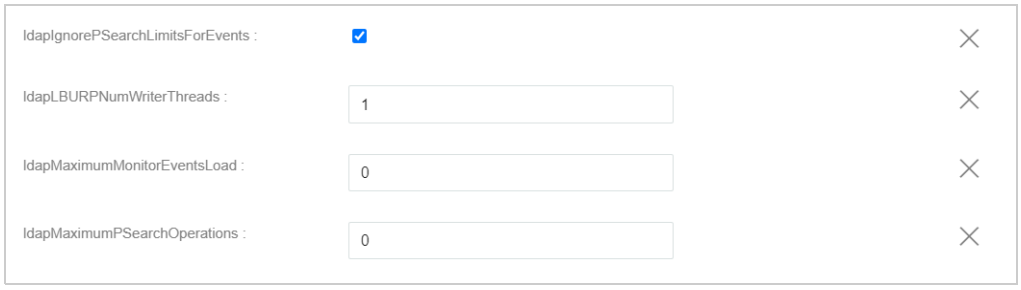
You can use OpenText Identity Console to view or edit persistent searches.

1. On the home page of OpenText Identity Console, click **LDAP Configuration** tile.
2. Click **LDAP server** object.
3. Click **Others** drop down menu.
4. By default, the **LdapEnablePSearch** check box is checked. To disable and prevent persistent searches on this server, uncheck the check box.

   NOTE: If you disable a previously established persistent search operation, the operation might continue even after this option is disabled and the server is refreshed.

5. Control the number of concurrent persistent searches on this server.

   Specify a value in the **LdapMaximumPSearchOperations** field. A value of zero allows unlimited concurrent persistent searches.



6. Control whether to ignore size and time limits.

   To control whether size and time limits should be ignored after the persistent search request has sent the initial search result set, select the **LdapIgnorePSearchLimitsForEvents** check box.

   If you don't select this option, the entire persistent search operation is subject to the search restrictions. If either limit is reached, the search will fail, with the appropriate error message.

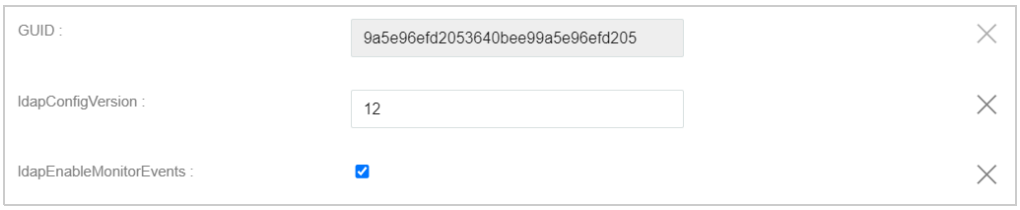
7. Click **Save**, then click **OK**.

Controlling Use of the Monitor Events Extended Operation

1. On the OpenText Identity Console home page, click **LDAP Configuration** tile.
2. Click **LDAP Servers** object.
3. Click **Others** drop down menu.
4. Control whether client applications can monitor events on this LDAP server.

To enable client applications to monitor events on this LDAP server, select the **LdapEnableMonitorEvents** check box.

To disable the monitoring of events, unselect the check box.



5 Control the maximum load that event monitoring applications can place on the server.

Enter a value in the **LdapMaximumMonitorEventsLoad** field.

Processing event data and sending event notifications to monitoring applications involves computational overhead on the LDAP server. For a given event, the exact load on the server depends on the frequency of the event being monitored, the data associated with the event, and the number of client applications monitoring the event.

The Maximum Event Monitoring Load is a relative value that reflects how much of a load the event monitoring extension is allowed to place on the server. A zero value indicates no limit. To find an appropriate value for this attribute, experiment.

6 Click **Save**, then click **OK**.

Getting Information about the LDAP Server

To get information about an LDAP server, you use ICE or an LDAP search. These utilities request information from rootDSE (Directory Service Agent, specific entry).

rootDSE is a pseudo object in a directory tree. It is an unnamed entry at the root of the tree. rootDSE holds information that is specific to the server that you are connected to. For example, rootDSE knows where the schema is located and the extensions and controls that the schema supports.

Because rootDSE is not a named entry in the tree, an LDAP server does not return rootDSE to the client as part of any normal search operation.

The following table lists information from rootDSE.

Information and Description	Excerpt
The schema's location: You find where the schema for the LDAP server or tree is located by reading the subschemaSubentry. For eDirectory, cn=schema is the base for the search.	subschemaSubentry: cn=schema

Information and Description	Excerpt
Supported extensions: Extensions enable you to manage the server (for example, creating or merging contexts, adding new replicas, refreshing the LDAP server, removing replicas, changing the replica type from master to read/write or read-only) and identities. Extensions are in ASN.1OID format. For names of extensions, see LDAP Extensions (https://www.novell.com/documentation/developer/ldapover/ldap_enu/data/a6ik7oi.html).	supportedExtension: 2.16.840.1.113719.1.27.100.12 supportedExtension: 2.16.840.1.113719.1.27.100.7 supportedExtension: 2.16.840.1.113719.1.27.100.8
Which vendor is providing the LDAP server.	vendorName: OpenText Corporation.
Which directory version the LDAP server supports.	vendorVersion: eDirectory v8.7.0 (10410.29)
Which version of OpenText eDirectory is running.	vendorVersion: eDirectory v8.7.0 (10410.29)
The directory server name and the directory tree name.	dsaName: cn=WestWindNDS,o=westwind directoryTreeName: t=WESTWINDTREE
Supported SASL mechanisms.	supported SASLMechanisms: EXTERNAL supported SASLMechanisms: DIGEST-MD5 supported SASLMechanisms: NMAS LOGIN
Which version of LDAP server is supported.	supportedLDAPVersion: 2 supportedLDAPVersion: 3
Server statistics: rootDSE provides a variety of statistics about the LDAP server (for example, the number of strong authentication binds).	errors: 0 securityErrors: 0 chainings: 3 referralsReturned: 6 extendedOps: 0 abandonOps: 0 wholeSubtreeSearchOps: 1

Information from rootDSE is useful for application developers.

Scenario: Developing an Application— Henri is writing an application that creates a new replica. Henri reads rootDSE and finds supportedExtension: 2.16.840.1.113719.1.27.100.7 in the list. Henri knows that the server supports the call to create a new replica.

Also, OpenText Identity Console checks to see what functionality is available in rootDSE and then behaves according to that information.

To search rootDSE, enter the following at a workstation:

Syntax:

```
ldapsearch -H ldap://hostname:port -b "" -s base "objectclass=*"
```

Example:

```
ldapsearch -H ldap://10.10.10.10:389 -b "" -s base "objectclass=*"
```

This search can be performed by any application using the ldap_search APIs.

The key to the search is that the scope is base (`-s base`). Also note that the base is null and the filter is set to `objectclass=*`. In the case of this client, the base is `-b`.

For more information on reading the rootDSE, refer to one of the following:

- LDAP Libraries for C (https://www.novell.com/documentation/developer/cldap/ldaplibc/data/a2etgcm.html)
- LDAP Classes for Java (https://www.novell.com/documentation/developer/jldap/jldapenu/data/bktitle.html)

For information on LDAP search filters, see LDAP Search Filters (https://www.novell.com/documentation/developer/ldapover/ldap_enu/data/a3saoeg.html). This section is in the LDAP and NDS Integration section of the NDK documentation.

Configuring Generalized Time Support

The Generalized Time Support option allows you to display time in the YYYYMMDDHHmmSS.0Z format. You can enable or disable LDAP Generalized Time support using the ldapconfig utility or the LDAP in OpenText Identity Console.

Generalized time support can be enabled using any one of the following methods:

LDAP Configuration in OpenText Identity Console

1. On the OpenText Identity Console home page, click the **LDAP Configuration** tile.
2. Click **LDAP Servers**.
3. Click **Others** drop down menu > **LdapGeneralizedTime**, click **Save** > then click **OK**.

ldapconfig Utility

```
ldapconfig set "ldapGeneralizedTime=yes/no" -a <admin-FDN> -w <admin-password">
```

Configuring Permissive Modify

The current LDAP modify operation can be extended by setting the ldapPermissiveModify option to TRUE. If you attempt to delete an attribute that does not exist or to add any value to an attribute that already exists, the operation goes through without displaying any error message.

LDAP Configuration in OpenText Identity Console

1. On the OpenText Identity Console, click **LDAP Configuration** tile.
2. Click **LDAP Servers**.
3. Click **Others** drop down menu > select **LdapPermissiveModify** check box, click **Save** then click **OK**.

ldapconfig Utility

```
ldapconfig set "ldapPermissiveModify=yes/no" -a <admin-FDN> -w <admin-password">
```

Proxied Authorization Control

OpenText eDirectory provides the flexibility of controlling proxied authorization through the LDAP protocol as specified in RFC 4370. The proxied authorization control allows a client to request that an operation be processed with a provided authorization identity instead of the current authorization identity associated with the connection. This feature provides a mechanism for specifying an authorization identity for each operation, which benefits the clients that need to perform several operations on behalf of multiple users.

To authenticate with the OpenText eDirectory server, an administrator must provide the proxied authorization control OID `2.16.840.1.113730.3.4.18` in the client request. To use the proxied authorization control, the authenticated user should have supervisor rights on the impersonated user.

1. Create an OpenText eDirectory tree and add user objects to it.
2. On OpenText Identity Console home page, click **Rights Management** tile.
3. Click **Trustees**.
4. Click **Search Object** and select user from the list.
5. On the **Context Browser** screen > select the User > click **OK**.
6. Click **Assigned Rights** > select the assigned rights for the selected user. Example: read, write, or nested.
7. Click **Done**, then click **Apply**.

To perform proxied authorization for `ldapsearch`, use the following command:

```
ldapsearch -x -H ldap://[ldap host ip]:[ldap-port] -D <Admin DN> -w
<Password> -e '!authzid=dn:<Impersonate user> -b o=novell -s one
```

To perform other LDAP operations using proxied authorization control, provide `2.16.840.1.113730.3.4.18` OID in the LDAP request.

Auditing the Proxied Authorization Operations

To audit the proxied authorization operations, OpenText eDirectory provides a new event called DSE_IMPERSONATE.

LDAP Paged Search Control

The LDAP paged search control is used with an LDAP Search operation to allow clients to receive search results in a controlled manner limited by the page size. The size of the page can be determined by the administrator based on the available resources and infrastructure. For more information about the LDAP Paged Search Control, see RFC 2696.

To use paged search control, an administrator must provide the paged search control OID `1.2.840.113556.1.4.319` in the search request. In order to use paged search control, an administrator must also ensure that all the partitions are present in the same server.

When all the partitions are not present in the same server and the administrator performs LDAP Search with paged control and criticality level set, the below error message will be displayed on the client side:

```
Server is unwilling to perform (53)
```

In case the administrator wants to avoid receiving the above error message, set the NDSD_NLDAP_IGNORE_CRITICALITY environment variable to true.

NOTE: Any version of OpenText eDirectory previous to 9.2 SP2 failed to display such error message on client side even if criticality level was set with paged search control.

To perform paged search control for ldapsearch, use the following command:

```
ldapsearch -x -H ldap://[ldap host ip]:[ldap-port] -D <Admin DN> -w
<Password> -E \!pr=<page size>/<prompt|no prompt> -b o=novell -s one
```

LDAP Extended DN Control

OpenText eDirectory provides the LDAP Extended DN Control which is used with an extended LDAP search to request an extended form of object Distinguished Name. The extended form includes a string representation of Object GUID along with Distinguished Name of the object.

To use the LDAP Extended DN Control feature with the eDirectory server, an administrator must provide the LDAP Extended DN Control OID 1.2.840.113556.1.4.529 in the extended LDAP search request.

The Extended DN Control enables the client to request that the results returned by an LDAP search that uses this control return the GUID data of an object along with the object distinguishedName, which is returned as follows:

<GUID=xxxxxxxx>;distinguishedName

Where xxxxxxxx is a string that contains the GUID, and distinguishedName is the DN, as in cn=users,dc=fabrikam,dc=com.

LDAP extended DN control can be passed along with an integer flag value. The flag value passed in specifies the string format of the returned GUID value, and is set to the following Ber-encoded sequence:

```
Sequence {
  Flag    INTEGER
}
```

A flag value 0 specifies that the GUID value be returned in hexadecimal string format such as <GUID=3BC72D2DEC5A704BBDC21F4EF97B7870>.

A flag value of 1 will return the GUID value in standard string format such as <GUID=098f2470-bae0-11cd-b579-08002b30bfeb>.

There are several complex data types of OpenText eDirectory that include DN as a part of it. OpenText eDirectory handles only the following complex data types with LDAP Extended DN Control:

- SYN_PATH (GUID is returned for volumeDN)
- SYN_DN
- SYN_TYPED_NAME

NOTE: LDAP search performance will be impacted while using the Extended DN Control with the above mentioned complex data types.

Examples:

The following C++ example code shows how to manually format the sequence data. The `ber_printf` function is used to create the sequence data. The flags portion contains the GUID string format specifier:

```
LDAPControl *FormatExtDNFlags(int iFlagValue)
{
  BerElement *pber = NULL;
  LDAPControl *pLControl = NULL;
  berval *pldctrl_value = NULL;
  int success = -1;

  // Ensure that iFlagValue is either 0 or 1. Convert TRUE (-1) to a legal
value.
  if(iFlagValue != 0)
         iFlagValue = 1;

  // Format and encode the SEQUENCE data in a BerElement.
  pber = ber_alloc_t(LBER_USE_DER);
  if(pber==NULL) return NULL;
  pLControl = new LDAPControl;
  if(pLControl==NULL) { ber_free(pber,1); return NULL; }
  ber_printf(pber,"{i}",iFlagValue);

  // Transfer encoded data into a BERVAL.
  success = ber_flatten(pber,&pldctrl_value);
  ber_free(pber,1);
  if(success != 0) {return NULL;}

  // Copy the BERVAL data to the LDAPControl structure.
  pLControl->ldctl_oid = LDAP_SERVER_EXTENDED_DN_OID;
  pLControl->ldctl_iscritical = true;
  pLControl->ldctl_value.bv_val = new char[pldctrl_value->bv_len];
  memcpy(pLControl->ldctl_value.bv_val,
         pldctrl_value->bv_val, pldctrl_value->bv_len);
  pLControl->ldctl_value.bv_len = pldctrl_value->bv_len;

  // Cleanup temporary berval.
  ber_bvfree(pldctrl_value);

  // Return formatted LDAPControl data.
  return pLControl;
}
```

The following C++ example code shows how to use the extended DN control with the `ldap_search_ext_s` function:

```
        int err;
        LDAP *ldapConnection = NULL;
        LDAPControl *pExtDNControl;
        LDAPControl *controlArray[2];
        LDAPMessage *results = NULL;
        LDAPMessage *message = NULL;
        char *dn = NULL;

        // Connect to the default LDAP server.
        ldapConnection = ldap_open( NULL, 0 );
        if ( ldapConnection == NULL ) go to FatalExit0;

        // Bind to the server using default credentials.
        err = ldap_simple_bind_s( ldapConnection, NULL, NULL);
        if (LDAP_SUCCESS != err) go to FatalExit0;

        // Setup the extended DN control, requesting 'standard string'
format.
        pExtDNControl = FormatExtDNFlags(1);
        if (pExtDNControl == NULL) go to FatalExit0;
        controlArray[0] = pExtDNControl;
        controlArray[1] = NULL;

        // Perform a synchronous search.
        err   = ldap_search_ext_s( ldapConnection,
                        "cn=users,dc=Fabrikam,dc=com",
                        LDAP_SCOPE_SUBTREE,
                        "objectClass=*",
                        NULL,          // Retrieve all attributes.
                        0,             // Retrieve attributes and values.
                        (LDAPControl **) &controlArray,
                        NULL,          // Client controls.
                        0,             // Timeout.
                        0,             // Sizelimit.
                        &results       // Receives identifier for results.
                        );
        if (LDAP_SUCCESS != err) go to FatalExit0;

        // Process the search results.
        message = ldap_first_entry( ldapConnection, results );
        while (message != NULL)
        {
```

```
                // Print the distinguished name of the object.
                dn = ldap_get_dn( ldapConnection, message );
                if (!dn) go to FatalExit0;
                printf( "  Distinguished Name is : %s\n", dn );
                ldap_memfree(dn);
                message = ldap_next_entry( ldapConnection, message );
        }

FatalExit0:
        if (ldapConnection)
                ldap_unbind( ldapConnection );
        if (results)
                ldap_msgfree( results );
}
```

Auditing LDAP Events

LDAP auditing enables the applications to monitor/audit LDAP operations such as Add, Modify, Search, and so on, and to fetch useful information from the LDAP server such as the connection information, the client IP to which the server was connected when LDAP operation happened, the message ID, the result code of the operation, and so on.

For more information on auditing LDAP events, refer to the LDAP Event Services (https://www.novell.com/documentation/developer/ldapover/ldap_enu/data/ag7bleo.html).

15 Backing Up and Restoring OpenText eDirectory

OpenText eDirectory is designed to provide fault tolerance for the tree through replication, so that if one server is not available, other servers can provide access. Replication is the primary method for protecting OpenText eDirectory.

Replication, however, is not possible in a single-server environment. Also, replication might not provide a complete restore of individual servers in case of a server hardware failure or other damage, or in the event of a disaster such as a fire or flood in which you lose multiple servers. Backing up OpenText eDirectory on each server increases the fault tolerance for your network.

The eDirectory Backup Tool enables you to back up the OpenText eDirectory database on your individual servers. It has the following benefits:

- **Same tool for all platforms.**
- **Provides hot continuous backup.** You can back up your server without closing the OpenText eDirectory database, and you still get a complete backup.
- **Supports a quick restore of an individual server.** This is especially helpful in the event of hardware failure.
- **Scalable.** You can back up a server whose OpenText eDirectory database contains tens or hundreds of millions of objects. The speed of the backup process is limited mainly by I/O channel bandwidth.
- **Can support a quick restore of the tree, when used with replica planning and DSMASTER servers.** Even without using DSMASTER servers, some level of recovery for the tree should be possible. See “Using DSMASTER Servers as Part of Disaster Recovery Planning” on page 430.
- **Lets you back up related files.** You can back up files on the server that are related to the database, such as NICI security files, stream files, and any files you specify (such as `autoexec.ncf`) in an include file.
- **Can restore OpenText eDirectory to the state it was in at the moment before it went down,** if you use continuous roll-forward logging. See “Using Roll-Forward Logs” on page 432.
- **Makes hardware upgrade simpler.** Doing a cold backup and then restoring the OpenText eDirectory database is an easy way to transfer the server's identity to a new machine or safeguard it while you make changes such as RAM upgrades. See “Upgrading Hardware or Replacing a Server” on page 541.
- **Works within the distributed nature of OpenText eDirectory.** You can ensure that a restored server matches the synchronization state that other servers in the tree expect by turning on continuous roll-forward logging.
- **Allows unattended backups.** You can create batch files to run unattended backups through the DSBK Client.

The OpenText eDirectory Backup Tool is designed to give you a complete backup and restore of the database and associated files on an individual server. It does not support backup and restore for individual objects or sections of the tree.

Also, it must be used in conjunction with file system backups to put the eDirectory backup files safely on tape.

For OES 2 Linux, you can back up OpenText eDirectory using OpenText Storage Management Services. SMS provides a target service agent (TSA) for backing up and restoring OpenText eDirectory. TSANDS services provide an implementation of the SMS APIs for the Directory trees. Applications can make use of this feature for backing up and restoring OpenText eDirectory objects.

TSANDS supports the following features that backup applications can take advantage of:

- Filters that can be applied to the OpenText eDirectory objects.
- Selective restores OpenText eDirectory objects from the backed up data.
- Ability to rename a particular set of resources.
- Support for incremental and differential backups based on the OpenText eDirectory modification date.
- Formats data in a SIDF and therefore any SIDF-compliant software can interpret the data.

For more information on TSANDS usage, refer to the TSANDS man page.

This chapter contains the following topics:

- Checklist for Backing Up eDirectory (page 418)
- Understanding Backup and Restore Services (page 421)
- Using Roll-Forward Logs (page 432)
- Preparing for a Restore (page 436)
- Using DSBK (page 439)
- Backing Up and Restoring NICI (page 454)
- Recovering the Database If Restore Verification Fails (page 456)
- Scenarios for Backup and Restore (page 460)
- Disaster Recovery Plan using DSBK (page 466)
- LDAP-Based Backup (page 468)
- eDirectory Backup with SMS (page 469)
- ndsbackup Utility (page 768)

Checklist for Backing Up OpenText eDirectory

To make sure objects in a multiple-server tree are accessible even if a server is down:

❒ For multiple-server trees, ensure that all OpenText eDirectory partitions are replicated on more than one server, for fault tolerance.

For information on creating replicas, see “Adding a Replica” on page 147.

To allow a quick and complete restore of individual servers (such as after a hardware failure):

- ❒ Do a full backup of the OpenText eDirectory database regularly (such as weekly).
- ❒ Do an incremental backup regularly (such as nightly).
- ❒ Ensure that you select NICI and stream attributes while taking a full backup of an EBA-enabled server. Otherwise, the backup operation fails.
- ❒ Do full and incremental tape backups of the file system shortly after full or incremental OpenText eDirectory database backups are completed.

  The Backup Tool writes the backup files to a directory you specify on the server, but has no way of placing the data directly to tape. File system backup should be set to run after the OpenText eDirectory backup has run, to place the database backup files on tape for safe storage.
- ❒ Turn on and configure roll-forward logging, if it's necessary in your environment.

  You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open. The restore by default won't open a database that shares replicas with other servers unless it is restored back to the state it was in at the moment before it went down.

  In a single-server environment, roll-forward logging is not required for the restore verification process, but you can use it if you want to be able to restore OpenText eDirectory to the moment before it went down instead of just to the last backup.

  Here is a list of the main issues you must address when you turn on roll-forward logging. For more information, see "Using Roll-Forward Logs" on page 432.

  - Specify a new location for the roll-forward logs (don't use the default).

    The logs must be local to the server. For fault tolerance, they must not be stored on the same disk partition/volume or the same storage device as OpenText eDirectory. You might want a separate disk partition/volume just for roll-forward logs.
  - Document where the roll-forward logs are placed, so that you can find them in the event of a failure.

    To find out the location when the server is healthy, refer to the "Location of the Roll-Forward Logs" on page 434. But, if the server has a failure that affects OpenText eDirectory (such as a hardware failure), you won't be able to look up the location of the roll-forward logs.
  - Monitor disk space on the disk partition/volume where the roll-forward logs are stored, so that you can prevent it from filling up.

    If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory will stop responding on that server.
  - Restrict access to where the roll-forward logs are kept, so that unauthorized users cannot see them.
  - If a restore is necessary, make sure you re-create the roll-forward log configuration on the server after the restore is complete. The settings are reset to the default during a restore. After turning on the roll-forward logs, you must also do a new full backup.
- ❒ If you use NICI, ensure that your OpenText eDirectory backups include NICI security files as eDirectory requires the same NICI files to open the DIB and read the encrypted data.

  For more information, see Backing Up and Restoring NICI in the NICI Administration Guide.

- ❒ For multiple-server trees, if you are using the Backup Tool to back up a server, you should upgrade all the servers that share replicas with it to eDirectory 8.5 or later.

  The restore verification process is backward compatible only with 8.5 or later. For more information about the restore verification, see “Overview of How the Backup Tool Does a Restore” on page 423.
- ❒ Periodically check the backup log file to make sure that unattended backups were successful.
- ❒ Remove the old log files from the Roll Forward Log directory.
- ❒ Do a cold backup before upgrading a server, as described in “Upgrading Hardware or Replacing a Server” on page 541.
- ❒ For multiple-server trees, ensure that all OpenText eDirectory partitions are replicated on more than one server, for fault tolerance.

  In addition to making objects available when a server is down, such as during maintenance, replicating your partitions also provides fault tolerance in a case where you lose a server, such as a hardware failure. If a server in a multiple-server tree holds a partition that is not replicated, and the server has a failure, there's a risk that you might not be able to recover the partition. It's best to make sure all partitions are replicated. For more information on why you might not be able to recover an unreplicated partition in a multiple-server tree, see Overview of How the Backup Tool Does a Restore, Using Roll-Forward Logs, and Recovering the Database If Restore Verification Fails.

  For information on replication, see “Replicas” on page 56 and Chapter 6, “Managing Partitions and Replicas,” on page 143.
- ❒ Ensure that the backup tapes containing the OpenText eDirectory and file system backups are in a safe location.
- ❒ Regularly test your backup strategy to make sure it meets your goals.
- ❒ (Optional) If you plan to access servers remotely to do cold backups (a full backup with the database closed) or to do advanced backup and restore tasks, install DSBK on the machine you plan to use. Also, arrange for access (such as VPN access) behind the firewall.

  DSBK is installed with OpenText eDirectory on the server, and you can also use it on workstations with Sun JVM 1.3.1. For information on installing and configuring DSBK, see “Using DSBK” on page 439.
- ❒ (Optional) If you plan to access servers remotely to do cold backups (a full backup with the database closed) or to do advanced backup and restore tasks, install eMBox on the machine you plan to use. Also, arrange for access (such as VPN access) behind the firewall.

  eMBox is installed with OpenText eDirectory on the server, and you can also use it on workstations with Sun JVM 1.3.1. For information on installing and configuring eMBox, refer to the “Using the eMBox Client for Backup and Restore” on page 570.

To prepare for a disaster in which you lose multiple servers:

- ❒ Address the issues listed above.
- ❒ For multiserver trees, consider creating DSMASTER servers to help you prepare for the event of a disaster.

  See “Using DSMASTER Servers as Part of Disaster Recovery Planning” on page 430.
- ❒ Regularly test your disaster recovery strategy to make sure it meets your goals.

Understanding Backup and Restore Services

- “About the OpenText eDirectory Backup Tool” on page 421
- “What's Different between Backup and Restore in DSBK and TSA for NDS Backup” on page 422
- “Overview of How the Backup Tool Does a Restore” on page 423
- “Format of the Backup File Header” on page 424
- “Format of the Backup Log File” on page 429
- “Using DSMASTER Servers as Part of Disaster Recovery Planning” on page 430
- “Transitive Vectors and the Restore Verification Process” on page 431

About the OpenText eDirectory Backup Tool

The Backup Tool provides hot continuous backup of the eDirectory database on an individual server. You can back up OpenText eDirectory on your server without closing the database, and you still get a complete backup that is a snapshot of the moment when the backup began. This feature means that you can create a backup at any time and OpenText eDirectory will be accessible throughout the process.

NOTE: Hot continuous backup is the default behavior—you can specify a “cold” backup with the database closed, if required.

The new backup also lets you turn on roll-forward logging to keep a record of transactions in the database since the last backup, so you can restore a server to the state it was in at the moment before it went down. You must turn on roll-forward logging for servers that participate in a replica ring, so that you can restore a server back to the synchronization state that the other servers expect. If you don't, when you try to restore from your backup files you will get errors and the database will not open. Roll-forward logging is off by default. For more information, see “Using Roll-Forward Logs” on page 432.

The Backup Tool does not back up all the objects in OpenText eDirectory at once, but only backs up the partitions on an individual server. This allows for better restore of an individual server and faster backups than the legacy TSA for NDS® backup. The legacy TSA for NDS backup still works as documented in eDirectory 8.6. Both the TSA for NDS and the new backup can be used if necessary. For a comparison, see “What's Different between Backup and Restore in DSBK and TSA for NDS Backup” on page 422.

The OpenText eDirectory Backup Tool must be used in conjunction with file system backups to put the OpenText eDirectory backup files safely on tape. OpenText has partnered with several leading providers of backup solutions. For a list, see OpenText eDirectory Partner Products (http://www.novell.com/partnerguide/section/466.html).

For a description of the format for the backup files and log files that the Backup Tool creates, see “Format of the Backup Log File” on page 429 and “Format of the Backup File Header” on page 424.

What's Different between Backup and Restore in DSBK and TSA for NDS Backup

In previous versions of OpenText eDirectory, backup and restore was focused on backing up the tree, object by object.

The Backup Tool in eDirectory 8.7 introduced a completely new focus and new architecture. It's server-centric, not tree-centric, and you back up the OpenText eDirectory database on each server individually. It's much faster than the legacy TSA for NDS backup.

The legacy TSA for NDS backup tool can still be used to back up the tree, although we encourage you to use the new backup.

For more comparison information, see the following table.

Issue	Legacy TSA for NDS Backup	Backup Tool "Hot Continuous Backup"
Focus	Designed to back up the tree, object by object.	Designed to back up the OpenText eDirectory database on each server individually. Fault tolerance for the whole tree should be provided primarily by replication, but backing up each server provides additional fault tolerance. When planning a restore strategy for the tree after a disaster in which many servers are lost, consider using DSMASTER servers and replica planning as outlined in "Using DSMASTER Servers as Part of Disaster Recovery Planning" on page 430.
Speed	N/A	Significantly improved. Speed is one of the most important features of the new Backup.
Where the backup is placed	Allows backup to be placed directly to tape.	Places the backup files on the file system. You must use a file system backup to put them on tape for safe storage.
Cross-platform	Performs differently on each platform.	Works the same way on each platform.

Issue	Legacy TSA for NDS Backup	Backup Tool "Hot Continuous Backup"
Ability to restore individual servers	Not designed to provide this.	Provides the ability to restore an individual server after a hard drive failure or to use Backup to move a server from one machine to another. Provides the option to use roll-forward logging so you can restore a server to the state it was in at the moment before it went down, so it is in the synchronization state expected by other servers in a replica ring. Has the ability to back up files related to OpenText eDirectory on an individual server. For example, you can back up and restore NICI files. You can also create your own list of related files to include with the backup.
Ability to restore NICI files for a server	Not designed to provide this.	Lets you back up and restore NICI files, so you can access encrypted data after a restore. This can save you a lot of time when restoring.
Roll-forward logging for an individual server	Not designed to provide this.	Lets you keep a record of transactions in the database since the last backup, so you can restore a server to the state it was in at the moment before it went down. In a multiple-server environment, this allows you to restore a server to the synchronization state that the other servers expect. Roll-forward logging is off by default. For more information, see "Using Roll-Forward Logs" on page 432.

Overview of How the Backup Tool Does a Restore

Before restoring, you need to collect all your backup files by following the instructions in "Preparing for a Restore" on page 436. When you direct the Backup Tool to begin the restore through DSBK, the process is done by the Backup Tool as follows:

1. The DS Agent is closed.
2. The active DIB (Data Information Base) set is switched from the DIB set named NDS to a new DIB set named RST.

   NOTE: The existing NDS database is left on the server. If the restore verification fails it will once again become the active DIB set.

3. The restore is performed, restoring to the DIB set named RST.
4. The DIB set is disabled.

   The login disabled attribute is set on the pseudo server, preventing the DS Agent from being able to open using this DIB set.
5. The roll-forward log settings are reset to the default. You can prevent this by using `-s` switch.

This means that after a restore, roll-forward logging on the server is always set to off, and the location of the roll-forward logs is reset to the default.

NOTE: If you want roll-forward logging turned on for this server, you must plan to re-create your configuration for roll-forward logging after a restore, to make sure it is turned on and the logs are being saved in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

---

6. Verification of the restored RST database is performed.

   The server attempts to verify the consistency of the data that has been restored. It does this by contacting every server that it shares a replica with and comparing the transitive vectors.

   The output from this verification process is printed in the log file.

   If the transitive vector on the remote server is ahead of the local vector, then data is missing from the restore, and the verification fails.

   Here is an example of the information that's recorded in the log file if verification fails for one of the replicas, showing the transitive vectors that were compared:

```
Server: \T=LONE_RANGER\O=novell\CN=CHIP
   Replica: \T=LONE_RANGER\O=novell
      Status: ERROR = -6034
         Local TV             Remote TV
         s3D35F377 r02 e002   s3D35F3C4 r02 e002
         s3D35F370 r01 e001   s3D35F370 r01 e001
         s3D35F363 r03 e001   s3D35F363 r03 e001
         s3D35F31E r04 e004   s3D35F372 r04 e002
         s3D35F2EE r05 e001   s3D35F2EE r05 e001
         s3D35F365 r06 e003   s3D35F365 r06 e003
```

   For more information, see “Transitive Vectors and the Restore Verification Process” on page 431.

7. If verification is successful, RST is renamed to NDS and the login disabled attribute is cleared so it becomes the active OpenText eDirectory database on the server. If verification fails, the RST DIB is not renamed, and the active DIB set is set back to NDS.

   If verification fails, see “Recovering the Database If Restore Verification Fails” on page 456 for how to recover the server.

NOTE: It's possible to force the RST database to be activated and unlocked using advanced restore options, but this is not recommended unless suggested by OpenText Support.

Format of the Backup File Header

The backup files contain a header that you can read to learn important information such as

- The filename of the backup file when it was created.

  This is helpful if the filename has been changed since the backup was created.

- The current roll-forward log at the time of this backup.

If this is the last backup in the set you are restoring from, such as the last incremental backup in a set of one full backup and three incremental backups, this helps you because it indicates the first roll-forward log that you need for a complete restore.

- The replicas this server held.

  This is helpful if you did not have the placement of your replicas documented. If you experienced a disaster in which many servers were lost, the list of replicas shown in the backup file header might help you decide which servers to restore first.

- The names of the files that were included in the backup as specified in a user include file.
- The number of files in the backup set for that backup.

The header of the backup file for each individual backup is in XML format. Immediately following the header is the backup data from the database in binary code.

NOTE: Because of the inclusion of binary data at the end of the file, parsing the file would give errors, but the XML header complies with XML standards.

In cases where the backup spanned more than one file, the header information is included in each file in the set.

WARNING: When opening a backup file, just view the header—make sure you don't try to save or modify the file, or it might become truncated. Most applications can't save the binary data correctly.

The following is the DTD for the XML header. The DTD is included as part of the header in the backup file as well, for your reference.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE backup [
<!ELEMENT backup (file|replica)*>
<!ELEMENT file (#PCDATA)>
<!ELEMENT replica EMPTY>
<!ATTLIST backup version CDATA #REQUIRED
    backup_type (full|incremental) #REQUIRED
    idtag CDATA #REQUIRED
    time CDATA #REQUIRED
    srvname CDATA #REQUIRED
    dsversion CDATA #REQUIRED
    compression CDATA "none"
    os CDATA #REQUIRED
    current_log CDATA #REQUIRED
    number_of_files CDATA #IMPLIED
    backup_file CDATA #REQUIRED
    incremental_file_ID CDATA #IMPLIED
    next_inc_file_ID CDATA #IMPLIED>
```

```
<!ATTLIST file size CDATA #REQUIRED
    name CDATA #REQUIRED
    encoding CDATA "base64"
    type (user|nici) #REQUIRED>
<!ATTLIST replica partition_DN CDATA #REQUIRED
    modification_time CDATA #REQUIRED
    replica_type (MASTER|SECONDARY|READONLY|SUBREF|
    SPARSE_WRITE|SPARSE_READ|Unknown) #REQUIRED
    replica_state (ON|NEW_REPLICA|DYING_REPLICA|LOCKED|
    CRT_0|CRT_1|TRANSITION_ON|DEAD_REPLICA|
    BEGIN_ADD|MASTER_START|MASTER_DONE|
    FEDERATED|SS_0|SS_1|JS_0|JS_1|MS_0|MS_1|
    Unknown) #REQUIRED>
]>
```

The following table explains the attributes in the DTD.

Attribute	Explanation
backup version	Version of the Backup tool.
backup backup_type	Type of backup being performed, either full or incremental. A cold backup is a full backup.
backup idtag	A GUID based on the time of backup. This helps in identifying the backup, even if the filename of the backup file is changed.
backup time	Date and time the backup was started.
backup srvname	Distinguished name of the server being backed up.
backup dsversion	OpenText eDirectory version running on the server.
backup compression	Whether the Backup Tool has used compression on the backup data. This only applies to the backup data. The header itself will never be compressed.
backup os	Operating system the backup was performed on. We recommend that you restore only to the same operating system.
backup current_log	First roll-forward log that is required when restoring this backup. This helps you collect the correct set of files for a restore.
backup number_of_files	Number of files in the backup set. This value appears only in the first backup file.
backup backup_file	Filename of the current backup. If the backup spans multiple files, then the header for each file will show the filename including a number appended to show its order in the set. For an example of the filenames in a set of backup files, see -s *file_size*.
backup incremental_file_ID	If this is an incremental backup, this attribute shows the ID of the incremental file.
backup next_inc_file_ID	The ID that the next incremental backup will have when it is created. This helps you collect the correct set of files for a restore.

Attribute	Explanation
file size	Size of the data between the <file> tags for this file.
file name	Name and location of the file when it was backed up.
file encoding	The encoding algorithm used on the file.
file type	Indicates whether the file is a NICI file or a user included file.
replica partition_DN	Distinguished name of the partition. This is helpful if you did not have the placement of your replicas documented. If you experienced a disaster in which many server were lost, the list of replicas shown in the backup file header might help you decide which servers to restore first.
replica modification_time	Transitive vector for this replica at the time of the backup.
replica replica_type	Type of replica, such as master or read-only.
replica_state	State of the replica at the time of the backup, such as On or New Replica.

The following is an example of a backup file header from a Windows server, with NICI security files included in the backup:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE backup [
<!ELEMENT backup (file|replica)*>
<!ELEMENT file (#PCDATA)>
<!ELEMENT replica EMPTY>
<!ATTLIST backup version CDATA #REQUIRED
    backup_type (full|incremental) #REQUIRED
    idtag CDATA #REQUIRED
    time CDATA #REQUIRED
    srvname CDATA #REQUIRED
    dsversion CDATA #REQUIRED
    compression CDATA "none"
    os CDATA #REQUIRED
    current_log CDATA #REQUIRED
    number_of_files CDATA #IMPLIED
    backup_file CDATA #REQUIRED
    incremental_file_ID CDATA #IMPLIED
    next_inc_file_ID CDATA #IMPLIED>
<!ATTLIST file size CDATA #REQUIRED
    name CDATA #REQUIRED
    encoding CDATA "base64"
    type (user|nici) #REQUIRED>
<!ATTLIST replica partition_DN CDATA #REQUIRED
    modification_time CDATA #REQUIRED
    replica_type (MASTER|SECONDARY|READONLY|SUBREF|
    SPARSE_WRITE|SPARSE_READ|Unknown) #REQUIRED
    replica_state (ON|NEW_REPLICA|DYING_REPLICA|LOCKED|
    CRT_0|CRT_1|TRANSITION_ON|DEAD_REPLICA|
    BEGIN_ADD|MASTER_START|MASTER_DONE|
    FEDERATED|SS_0|SS_1|JS_0|JS_1|MS_0|MS_1|
```

```
    Unknown) #REQUIRED>
]>

<backup version="2" backup_type="full" idtag="3D611DA2" time="2002-8-
19'T10:32:35" srvname="\T=MY_TREE\O=novell\CN=DSUTIL-DELL-NDS"
dsversion="1041081" compression="none" os="windows"
current_log="00000003.log" next_inc_file_ID="2" number_of_files="0000001"
backup_file="c:\backup\header.bak"><replica partition_DN="\T=MY_TREE"
modification_time="s3D611D95_r1_e2" replica_type="MASTER"
replica_state="ON" /><replica partition_DN="\T=MY_TREE\O=part1"
modification_time="s3D611D95_r1_e2" replica_type="MASTER"
replica_state="ON" /><replica partition_DN="\T=MY_TREE\O=part2"
modification_time="s3D611D95_r1_e2" replica_type="MASTER"
replica_state="ON" /><replica partition_DN="\T=MY_TREE\O=part3"
modification_time="s3D611D96_r1_e2" replica_type="MASTER"
replica_state="ON" /><file size="190"
name="C:\WINDOWS\system32\novell\nici\bhawkins\XARCHIVE.001"
encoding="base64" type="nici">the data is included here</file>

<file size="4228"
name="C:\WINDOWS\system32\novell\nici\bhawkins\XMGRCFG.KS2"
encoding="base64" type="nici">the data is included here</file>

<file size="168"
name="C:\WINDOWS\system32\novell\nici\bhawkins\XMGRCFG.KS3"
encoding="base64" type="nici">the data is included here</file>

<file size="aaac" name="C:\WINDOWS\system32\novell\nici\nicintacl.exe"
encoding="base64" type="nici">the data is included here</file>

<file size="150" name="C:\WINDOWS\system32\novell\nici\NICISDI.KEY"
encoding="base64" type="nici">the data is included here
</file>

<file size="4228"
name="C:\WINDOWS\system32\novell\nici\system\Xmgrcfg.ks2"
encoding="base64" type="nici">the data is included here
</file>

<file size="168" name="C:\WINDOWS\system32\novell\nici\system\Xmgrcfg.ks3"
encoding="base64" type="nici">the data is included here
</file>

<file size="1414" name="C:\WINDOWS\system32\novell\nici\xmgrcfg.wks"
encoding="base64" type="nici">the data is included here
</file>

</backup>
```

After the header, the binary data for the backup of the database is included in the backup file.

Format of the Backup Log File

The OpenText eDirectory Backup Tool keeps a log that shows a high-level view of Backup Tool activity, containing information about previous backups. The log file contains a history of all backups, records backup start time and end time, and contains information about possible errors during the backup process. This file is appended with each backup. It is also placed in a location you specify.

It is useful for reviewing whether unattended backups were successful. The success or failure and the error code are displayed on the last line.

The Backup Tool log file also gives the ID of backups that have been done, which helps you gather the correct set of full and incremental backup files for a restore. The first four lines are duplicates of information in the header of the backup file.

Also recorded in the log file are other files that were included in the backup of the database, such as NICI files or the files you specified in an include file.

For a restore, it will record the included files that were restored.

The following are two examples of log file entries:

```
|==================DSBackup Log: Backup================|
Backup type: Full
Log file name: sys:/backup/backup.log
Backup started: 2002-6-21'T19:53:5GMT
Backup file name: sys:/backup/backup.bak
Server name: \T=VIRTUALNW_TREE\O=novell\CN=VIRTUALNW
Current Roll Forward Log: 00000001.log
DS Version: 1041072
Backup ID: 3D138421
Backing up security file: sys:/system/nici/INITNICI.LOG
Backing up security file: sys:/system/nici/NICISDI.KEY
Backing up security file: sys:/system/nici/XARCHIVE.000
Backing up security file: sys:/system/nici/XARCHIVE.001
Backing up security file: sys:/system/nici/XMGRCFG.KS2
Backing up security file: sys:/system/nici/XMGRCFG.KS3
Backing up security file: sys:/system/nici/XMGRCFG.NIF
Starting database backup...
Database backup finished
Completion time 00:00:03
Backup completed successfully

|==================DSBackup Log: Restore================|
Log file name: sys:/save/doc.log
```

```
Restore started: 2002-7-19'T19:1:34GMT
Restore file name: sys:/backup/backup.bak
Starting database restore...
Restoring file sys:/backup/backup.bak
Restoring file sys:/system/nici/INITNICI.LOG
Restoring file sys:/system/nici/NICISDI.KEY
Restoring file sys:/system/nici/XARCHIVE.000
Restoring file sys:/system/nici/XARCHIVE.001
Restoring file sys:/system/nici/XMGRCFG.KS2
Restoring file sys:/system/nici/XMGRCFG.KS3
Restoring file sys:/system/nici/XMGRCFG.NIF
Database restore finished
Completion time 00:00:15
Restore completed successfully
```

Using DSMASTER Servers as Part of Disaster Recovery Planning

If you have a multiple-server environment and want to plan for recovery after a disaster in which all your servers are lost, you can use DSMASTER servers as part of the plan for your tree.

The Backup Tool is used to back up each server separately. It is server-centric, not tree-centric. However, if you create DSMASTER servers, you can use Backup Tool functionality specifically to back up your whole tree structure. An example of this strategy is outlined in “Scenario: Losing All Servers in a Multiple-Server Environment” on page 464.

When restoring after a disaster, one of the main concerns is how to avoid restoring replicas of the same partition that are inconsistent with each other. If you lose roll-forward logs for your servers as part of a disaster, you won't be able to restore all your servers to the same moment in time. Without the roll-forward logs, the replicas you have in your backups are inconsistent with each other and would cause problems if they were all restored and brought into the tree together.

NOTE: The restore verification process is designed to help prevent these problems. By default, a restored OpenText eDirectory database will not open after the restore if it is inconsistent with the other replicas.

You can use DSMASTER servers to help you prepare for this issue, by creating a master copy of your tree that you could use as a starting point.

To use DSMASTER servers to help prepare for a disaster:

- Plan your replicas so that you have one server that contains a replica of every partition in your tree, so a copy of the whole tree is in the OpenText eDirectory database on one server (or, if your tree is large, you can use a couple of key servers). This kind of server is often called a DSMASTER server. The replicas on the DSMASTER server should be master or read/write replicas.

  NOTE: If a couple of key DSMASTER servers are used instead of just one, keep in mind that ideally each DSMASTER server should have a unique set of replicas of partitions. There should be no overlap between them, to avoid inconsistencies between the replicas when restoring after a disaster.

If your servers were lost in a disaster, you would not have access to the most recent roll-forward logs for restoring because roll-forward logs are saved locally on the server, so all the DSMASTER servers probably could not be restored to the same moment in time. If the same replica were held on two DSMASTER servers, the two copies would probably not be identical and would cause inconsistencies in the tree. So, for disaster recovery planning it's best to not have the same partition replicated on more than one DSMASTER server.

For general information on replicas, see “Replicas” on page 56.

- Back up these DSMASTER servers regularly to create a backup copy of your tree. You might want to take extra precautions for storing the backups of DSMASTER servers as part of your disaster recovery plan.

If your tree is designed this way, in the event of a disaster you could get your tree structure up and running again quickly by restoring just that one server (or small group of key servers) and making sure the replicas it holds are designated as the master replicas.

After your tree structure is responding again, you could then move to the task of restoring other servers that were lost, using just the full and incremental backup files. Because you don't have the roll-forward logs, the verification of the restore process will fail for these other servers. To bring them back into the tree, you would remove them from the replica ring, change all their replica information to external references using DSRepair, and then re-add the replicas to the servers using replication from the copy on the DSMASTER server. These steps are documented in “Recovering the Database If Restore Verification Fails” on page 456.

If a disaster occurs in which you lose many servers but not all, the issues with replicas will probably be complex, and you should contact OpenText Support.

Transitive Vectors and the Restore Verification Process

A transitive vector is a time stamp for a replica. It is made up of a representation of the number of seconds since a common specific point in history (January 1, 1970), the replica number, and the current event number. Here's an example:

```
s3D35F377 r02 e002
```

In the context of backup and restore, it's important because the transitive vector is used to verify that the server restored is in sync with the replica ring it participates in.

Servers that hold replicas of the same partition communicate with each other to keep the replicas synchronized. Each time a server communicates with another server in the replica ring, it keeps a record of the transitive vector the other server had when they communicated. These transitive vectors allow the servers in a replica ring to know what information needs to be sent to each replica in the ring to keep all the replicas synchronized. When a server goes down, it stops communicating, and the other servers don't send updates or change the transitive vector they have recorded for that server until the server starts communicating again.

When you restore eDirectory on a server, the restore verification process compares the transitive vector of the server being restored to the other servers in the replica ring. This is done to make sure that the replicas being restored are in the same state that the other servers expect.

If the transitive vector on the remote server is ahead of the local vector, then data is missing from the restore, and the verification fails. For example, data might be missing because you did not turn on continuous roll-forward logging before the last full or incremental backup, you did not include the roll-forward logs in the restore, or the set of roll-forward logs you provided for the restore was not complete.

By default the restored OpenText eDirectory database is not opened if it is inconsistent with the other replicas.

For an example of the log file entry when transitive vectors don't match, see “Overview of How the Backup Tool Does a Restore” on page 423.

For information on what to do if the restore verification fails, see “Recovering the Database If Restore Verification Fails” on page 456.

Using Roll-Forward Logs

Roll-forward logging is similar to journaling on other database products. The roll-forward logs (RFLs) are a record of all changes to the database.

The advantage of using roll-forward logging is that the roll-forward logs give you a history of changes since the last full or incremental backup, so you can restore OpenText eDirectory to the state it was in at the moment before a failure. Without roll-forward logs, you can restore OpenText eDirectory only to the point of the last full or incremental backup.

OpenText eDirectory creates a record of transactions in a log file before committing them to the database. By default, the log file for these records is reused over and over (consuming only a small amount of disk space), and the history of changes to the OpenText eDirectory database is not being saved.

When you turn on continuous roll-forward logging, the history of changes is saved in a set of consecutive roll-forward log files. Roll-forward logging does not reduce server performance, but simply saves the log file entries that OpenText eDirectory is already creating.

You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open. The restore by default won't open a database that shares replicas with other servers unless it is restored back to the state it was in at the moment before it went down. If you don't have roll-forward logs, you must follow a separate procedure to try to recover, described in “Recovering the Database If Restore Verification Fails” on page 456.

Roll-forward logging is off by default. You must turn it on if you want to use it on a server. Roll-forward logging is also turned off and the settings returned to default when you restore a server, so after a restore you must turn it on again, re-create your configuration, and create a new full backup.

NOTE: The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

In a single-server environment, roll-forward logging is not required, but you can use it if you want to be able to restore OpenText eDirectory to the moment before it went down instead of just to the last backup.

Make sure you monitor disk space when roll-forward logging is on. For more information, see “Backing Up and Removing Roll-Forward Logs” on page 435.

In this section:

- “Issues to Be Aware of When Turning On Roll-Forward Logging” on page 433
- “Location of the Roll-Forward Logs” on page 434
- “Backing Up and Removing Roll-Forward Logs” on page 435
- “Cautionary Note: Removing OpenText eDirectory Also Removes the Roll-Forward Logs” on page 436

You can turn on and configure roll-forward logging using DSBK. See or “Configuring Roll-Forward Logs with DSBK” on page 443.

Issues to Be Aware of When Turning On Roll-Forward Logging

If you decide to use continuous roll-forward logging, you must be aware of the following issues:

- **Turn on roll-forward logging before a backup is done** if you want to be able to use this feature for restoring the database.
- **For fault tolerance, make sure that the roll-forward logs are placed on a different storage device than eDirectory.** For security, you should also restrict user rights to the logs. For more information, see “Location of the Roll-Forward Logs” on page 434.
- **Document the location of the roll-forward logs.** For more information, see “Location of the Roll-Forward Logs” on page 434.
- **Monitor the available disk space where the logs are located.** For more information, see “Backing Up and Removing Roll-Forward Logs” on page 435.
- **If the logs are turned off or lost, turn them back on, then do a new full backup** to ensure that you can make a full recovery. This is necessary in these cases:
  - After a restore. Roll-forward logging is turned off and the settings are reset to the default as part of the restore process.
  - If you lose the directory containing the roll-forward logs because of a storage device failure or other failure.
  - If roll-forward logs are unintentionally turned off.
- **If you turn on logging of stream files, the roll-forward logs use up disk space more quickly.**

  When logging of stream files (such as login scripts) is turned on, the whole stream file is copied into the roll-forward log every time there is a change. You can slow the growth of the log files by turning off roll-forward logging of stream files and, instead, back them up only when you do an incremental or full backup.
- **The slowest part of restoring the database is replaying the roll-forward logs.** Roll-forward logs grow larger based on how many changes are made to the tree structure and whether stream files (such as login scripts) are being logged.

  If your database changes often, you might need to consider more frequent OpenText eDirectory backups so that fewer changes need to be replayed from roll-forward logs during a restore.
- **Don't change the name of a roll-forward log file.** If the filename is different than when the log was created, the log file can't be used in a restore.

- **Keep in mind that removing OpenText eDirectory also removes all the roll-forward logs.** If you want to be able to use the logs for restoring in the future, before removing OpenText eDirectory you must first copy the roll-forward logs to another location.
- **If a restore is necessary, make sure you re-create the roll-forward logs configuration on the server after the restore is complete** to make sure they are turned on and are placed in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

  This step is necessary because during a restore, the configuration for roll-forward logging is set back to the default, which means that roll-forward logging is turned off and the location is set back to the default. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

Location of the Roll-Forward Logs

If you turn on roll-forward logging, you should change the location of the roll-forward log directory to a different storage device than OpenText eDirectory.

Here are the important issues to consider when choosing the location:

- **Don't leave them in the default location—make sure you put them on a different storage device than OpenText eDirectory.** This way, if OpenText eDirectory is lost because of a storage device failure, you can still access the roll-forward logs to restore OpenText eDirectory.

  If you only have one storage device on your server, the roll-forward logs can't provide fault tolerance for OpenText eDirectory in case of a storage device failure. In this case, you probably should not use the roll-forward logs.

  You can change the location of the roll-forward logs using Backup Configuration in setconfig DSBK. The roll-forward logs directory must be local to the server.
- **Document the location.** Document where the roll-forward logs are placed so that you can find them when you need to restore the database on a server. It's important to do this while the server is healthy, before any failures happen.

  To find out the location when the server is healthy, you can look it up in DSBK using the `backup getconfig` option. But if the server has a failure that affects eDirectory (such as hardware failure), you won't be able to look up the location of the roll-forward logs.

  If the server has already had a failure and you are trying to restore it, keep in mind that any new installations of OpenText eDirectory will show the default location of the roll-forward logs. So, if you have just reinstalled OpenText eDirectory as the first step of a restore process, OpenText eDirectory will not show the correct location of the roll-forward logs on the server before it went down. You will need to refer to your documentation to find out where they are.

  The settings for the roll-forward logs are also recorded in the `_ndsdb.ini` file, but that file is on the same disk partition/volume as OpenText eDirectory, so if you were to lose the storage device where OpenText eDirectory was located, you couldn't use the `_ndsdb.ini` file to look up the location.
- **Restrict rights to where the roll-forward logs are located.** This is a security issue. The information is not easily readable, but the logs could be decoded to reveal sensitive data.
- **Monitor the amount of free disk space to make sure there is enough.** See "Backing Up and Removing Roll-Forward Logs" on page 435.
- **A good strategy is to set up a disk partition/volume solely for the roll-forward logs.** This way, disk space and security privileges can be easily monitored.

- **The last directory in the path is created by OpenText eDirectory.** It is based on the name of the current OpenText eDirectory database.

  For example, if the location you specified was `d:\Novell\NDS\DIBFiles` and your OpenText eDirectory database was currently named NDS, the location of the roll-forward logs would be `d:\Novell\NDS\DIBFiles\nds.rfl`. If you renamed the database from NDS to ND1, the roll-forward log directory would be changed to `d:\Novell\NDS\DIBFiles\nd1.rfl`.

  When you change the location, the new directory is created immediately, but a roll-forward log is not created there until a transaction takes place in the database.

- **When restoring, all the necessary roll-forward logs must be in the same directory.** For more information, see "Preparing for a Restore" on page 436.

Backing Up and Removing Roll-Forward Logs

If left unchecked, roll-forward logs can fill up the disk partition/volume where they are placed. If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory stops responding on that server. We recommend that you periodically back up the log files and remove unused logs from the server to free up disk space.

To identify, back up, and remove roll-forward logs that are safe to remove:

1. Make a note of the name of the last unused roll-forward log.

   You can find out the name of the last unused roll-forward log in the DSBK Client, enter the `getconfig backup` command. See "Configuring Roll-Forward Logs with DSBK" on page 443 for instructions.

   The last unused roll-forward log is the most recent roll-forward log that the database has completed and is no longer using to record transactions. It's called the last unused roll-forward log because the database has finished writing to it and has begun a new log file, so it does not need to have this one open any more. The current roll-forward log in which the database is recording transactions is in use and is still needed by the database.

2. Do a file system backup of the roll-forward logs, to put them all safely on tape.
3. Remove the roll-forward logs that are older than the last unused roll-forward log.

WARNING: Keep in mind that you must be cautious when removing roll-forward logs from the server. Compare carefully with your tape backup to make sure you have a backup copy of everything you delete.

The last unused roll-forward log indicates which file the database has just completed and closed. It does not indicate whether it's safe to remove that file from the server. You must make sure that you remove only files that you have a tape backup for.

If you need to retrieve any of the roll-forward logs from tape for use in a restore because you have placed some of them on tape backup, keep in mind the following issues:

- As with any roll-forward logs used for a restore, log files retrieved from file system backup tapes must be placed in the same folder as the other roll-forward logs, local to the server being restored.
- You must compare time stamps for any files that are duplicated on the tape and on the server. Use the latest one, the one on the server, if the time stamps are not the same. For example, the roll-forward log file that was in use by the database during the time of the file system backup will be incomplete on the tape. The latest and complete version of that file will be on the server.

Cautionary Note: Removing OpenText eDirectory Also Removes the Roll-Forward Logs

If you remove OpenText eDirectory from your server, the roll-forward log directory and all the logs in it are also removed. If you want to be able to use the logs for restoring the server in the future, before removing OpenText eDirectory you must first copy the roll-forward logs to another location.

Preparing for a Restore

The most important part of restoring the OpenText eDirectory database is making sure it is complete. Before restoring an OpenText eDirectory database to a server, ensure the prerequisites have been met as described in "Prerequisites for Restoring" on page 436. If you are not sure how to gather the right backup files, see "Locating the Right Backup Files for a Restore" on page 437.

Prerequisites for Restoring

- ❒ All servers that share a replica with the server to be restored are up and communicating. This allows the restore verification process to check with servers that participate in the same replica ring.
- ❒ You have gathered all the backup files you need:
  - The full backup and subsequent incremental backup files are copied to one directory on the server to be restored.
  - All roll-forward logs since the last backup are in one directory on the server to be restored.

    If this server participates in a replica ring, you must make sure all the roll-forward logs created since the last backup are in one directory on the server, with the same filenames they had when they were created.

  See "Locating the Right Backup Files for a Restore" on page 437.

  NOTE: If you do not have backup files for the server, use Xbrowse to query OpenText eDirectory to help you recover server information. You must do this before you remove the Server object or any associated objects from the tree.

  Xbrowse and additional information is available from the OpenText Support Web site.

- ❒ You have installed OpenText eDirectory, in a new temporary tree.

You bring up the server in a new tree at first because you will create the server with the same name it had before the failure, and you don't want to cause confusion in the original tree by putting the newly installed server in the tree before the restore has re-created the server's complete identity. Completing the restore process for the database will put the server back into its original tree.

- ❒ (Conditional) If you are using roll-forward logging on this server, plan to re-create your configuration for roll-forward logging after the restore, to make sure it is turned on and the logs are being saved in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

  The restore process turns off roll-forward logging and resets the configuration for roll-forward logging back to the default.

  The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

- ❒ (Conditional) If any applications or objects need to find this server by its IP address, use the same IP address for the restored server.

During the restore process, the OpenText eDirectory Backup Tool first restores the full backup. After this is complete, the Backup Tool prompts you to enter the filenames of the incremental files. It provides you with the ID of the next file. After all incremental files are restored, the Backup Tool moves on to the roll-forward logs. See also “Overview of How the Backup Tool Does a Restore” on page 423.

After you have gathered all the files, perform the restore using the DSBK Client. See “Restoring from Backup Files with DSBK” on page 444.

Locating the Right Backup Files for a Restore

1. From your file system backup tape, copy the OpenText eDirectory full backup files to one directory on the server.

   You can check the Backup Tool log file if you want to confirm the ID of the last full backup.

2. From your file system backup tape, also copy each of the subsequent incremental backup files to the a directory on the server.

   To confirm that you have the right incremental backup files, look in the header of the full backup file. It contains the ID of the next incremental backup file, shown in the `next_inc_file_ID` attribute. The `next_inc_file_ID` is the same as the ID noted in the header of the incremental backup file in the `incremental_file_number` attribute. For a description of the header, see “Format of the Backup File Header” on page 424.

   WARNING: When opening a backup file, just view the header—make sure you don't try to save or modify the file, or it might become truncated. Most applications can't save the binary data correctly.

   Each incremental backup file will also contain the ID for the next incremental backup file.

   You can also look for the incremental backup ID in the Backup Tool log file.

The IDs are important because your backup files might have had the same filenames when they were created (for example, if you used the same batch file for unattended incremental backups so the backup filename specified was always the same), and you might have to change the filenames so you can place all the backups in the same directory. The ID in the header lets you find the correct files even if you have changed the filenames.

3 (Conditional) If you are using roll-forward logging on this server, make sure the roll-forward logs created since the last backup are in one directory on the server, with the same filenames they had when they were created.

If this server participates in a replica ring, you must restore using all the roll-forward logs. If you don't include all the roll-forward logs, the restore verification process will not be successful because the transitive vectors will not match when compared to the other replicas in the ring. By default the restored OpenText eDirectory database will not open after the restore if it is inconsistent with the other replicas.

Identify the first roll-forward log you need by opening the last backup file in a text editor and reading the current_log attribute in the header. You will need to collect this one and all the subsequent roll-forward logs.

WARNING: When opening a backup file, just view the header—make sure you don't try to save or modify the file, or it might become truncated. Most applications can't save the binary data correctly.

The roll-forward logs you need might not all be in the same location at the time you want to use them for a restore, so you need to make sure you have collected a complete set and placed them all in the same directory. The roll-forward logs might be in multiple locations for the following reasons:

- You have changed the location of the roll-forward logs directory since the last full or incremental backup.
- You have backed them up to tape using file system backup and then have removed them from the server, to save disk space.

  If you need to retrieve any of the roll-forward logs from tape backup, make sure you have the most current set. You must compare time stamps for any files that are duplicated on the tape and on the server. The roll-forward log file that was in use by the database during the time of the file system backup will be incomplete on the tape. The latest and complete version of that file will be on the server.
- You have changed the name of the OpenText eDirectory database since the last backup (such as from NDS to ND1). This changes the last directory name in the path to the roll-forward logs.

  For example, if the location you specified was `d:\novell\nds\dibfiles\`, and the name of your OpenText eDirectory database was NDS, the location of the roll-forward logs would be `d:\novell\nds\dibfiles\nds.rfl\`. If you renamed the database from NDS to ND1, the roll-forward log directory would change to `d:\novell\nds\dibfiles\nd1.rfl\`.

IMPORTANT: You must ensure that you provide all the necessary roll-forward logs. The Backup Tool cannot tell whether your set of roll-forward logs is complete. It will open and use the roll-forward logs in order. When it cannot find the next roll-forward log in the directory you specified, it ends the restore process. If you have not provided all the necessary roll-forward logs, the restore will be incomplete.

Using DSBK

DSBK is a thin command line parser that performs OpenText eDirectory Backup, and lets you to initiate a backup from the server console without having to log in first or set up Role-Based Services. It runs as a script on Linux and a console utility on Windows.

After a DSBK operation has completed, the results of the operation are written to a file (`dsbk.pipe` on Linux) that you can programmatically open and view the results. The first four bytes of this file contain error codes if any are generated during the operation. If there are no errors, the first four bytes of this file will contain zeros.

NOTE: Ensure that you have gone through all the guidelines given by OpenText before finalizing on your backup/restore setup.

Before performing backup and restore tasks, review Checklist for Backing Up OpenText eDirectory for an overview of the issues involved in planning an effective eDirectory backup strategy.

This section covers the following:

- "Prerequisites" on page 439
- "Using DSBK on Various Platforms" on page 440
- "Backing Up Manually with DSBK" on page 442
- "Automating the Backing Up of OpenText eDirectory" on page 443
- "Configuring Roll-Forward Logs with DSBK" on page 443
- "Restoring from Backup Files with DSBK" on page 444
- "Backup and Restore Command Line Options" on page 446
- "Running DSBK as a cron Job" on page 454

Prerequisites

- ❒ If you are planning to use roll-forward logs for this server, make sure they are turned on before a backup is made.

  You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open.

  For more information on roll-forward logs, see Using Roll-Forward Logs. For how to turn them on, see Configuring Roll-Forward Logs with DSBK.

- ❒ Decide which additional files you want to back up along with OpenText eDirectory, and create an include file if necessary.

  You can back up the stream files using switches. We recommend that you always back up NICI files. For more information on how to back up NICI, refer to Backing Up and Restoring NICI.

  If you want to include other files, such as the `autoexec.ncf` file, you must put the paths and filenames in an include file. Separate the paths and filenames with a semicolon and don't include hard returns or spaces. For example,
  `sys:\system\autoexec.ncf;sys:\etc\hosts;`.

- ❒ Plan to do a file system shortly after doing the eDirectory backup, to place the OpenText eDirectory backup files safely on tape. The Backup Tool only places them on the server.

  TIP: To make it easier to move the backup files to another storage device, you can specify the maximum size of eDirectory backup files as part of the `backup` command, using the `-s` option and a number for size in bytes. You can also use a third-party file compression tool on the files after they are created. They compress approximately 80%.

- ❒ Review the description of the command line options in Backup and Restore Command Line Options.

Using DSBK on Various Platforms

- "Using DSBK on Linux" on page 440
- "Using DSBK on Windows" on page 441

Using DSBK on Linux

DSBK commands can be run directly on shell of a Linux server where OpenText eDirectory is installed.

The output for the command is written into the OpenText eDirectory instance specific log file (Default instance: `/var/opt/novell/eDirectory/log/ndsd.log`):

```
DSBK HELP
To get help on a specific function type "help <function name>"
 Current functions:
  backup
  restore
  restadv
  getconfig
  setconfig
  cancel
```

DSBK commands can be entered into a `crontab` to execute `dsbk getconfig` and `dsbk backup` commands on a regular basis, allowing for full backups once in a week and incremental on other days, or whatever combinations are desired.

Using RFL in DSBK

- Turn on the RFL using the following command:

  ```
  dsbk setconfig -L
  ```

  The `-L` option starts a new roll forward logging session.

- Set a location for the roll-forward logs to be created using the following command:

  ```
  dsbk setconfig -L -r <roll forward log directory>
  ```

- Get a location for the roll-forward logs to be created using the following command:

  ```
  dsbk getconfig
  ```

- Remove the old log files from the roll-forward log directory during a backup using the `-a` option:

  ```
  backup -f <file name> -l <file name> [-s <size>] [-u <file name>] [-e
  <password>] [-t] [-w] [-a]
  [-b|-i|-c] [-o] [-d] [--config-file <configuration file>]
  ```

TIP: When using the DSBK utility interactively, have a second terminal window open with `tail -f <instance specific ndsd.log>` running so that the output to the entered commands is immediately readable.

Once the back up is completed, back it up using standard filesystem backup utilities.

NOTE: For detailed information on DSBK command line options, refer to the Backup and Restore Command Line Options.

Using DSBK on Windows

This section discusses the basic operation of the DSBK utility on the Windows platform.

For using DSBK on a Windows server that hosts OpenText eDirectory, perform the following steps:

1. Invoke the utility through the **OpenText eDirectory Services** console. **dsbk.dlm** is one of the options available in the list of services in the Services tab. The **dsbk** subcommand and any parameters for that subcommand are specified in the **Startup Parameters** field.
2. View the current configuration for the backup using the `getconfig` switch. The output of all the DSBK commands is appended to the `backup.out` file located in the OpenText eDirectory installation folder on Windows.
3. Set a location for the roll-forward logs to be created using the following command:

   ```
   setconfig -r <roll forward log directory> -L
   ```

   The `-L` option starts a new roll forward logging session.
4. Start backup on the tree by giving the following command:

   ```
   backup -f <backup file> -l <logfile> -t -w -b -e <password>
   ```

   Use the following options:

   - `-t`: Takes the backup of stream files.
   - `-w`: Overwrites any existing backup file with same name.
   - `-b`: Performs a full backup.
   - `-e <password>`: Performs a NICI backup using the password provided.
   - `-a`: Removes the old log files from the roll-forward log directory during a hot continuous backup.

   For example, start the backup as follows:

   ```
   backup -f c:\dsbk.bak -l c:\backup.log -t -w -b -e novell
   ```

   You can confirm the status of the backup done in the `backup.out` file.

NOTE: For detailed information on DSBK command line options, refer to the Backup and Restore Command Line Options.

You can turn on the RFL using the following command:

```
setconfig -r <roll forward log directory> -L
```

Backing Up Manually with DSBK

Use DSBK to back up data from an OpenText eDirectory database to a file you specify on the server where the backup is being performed. This backup file or set of files contains information necessary to restore OpenText eDirectory to the state it was in at the time of the backup. The results of the backup process are written to the log file you specify.

Using DSBK, you can do tasks such as the following:

- Do a full or incremental backup while the database is open (hot continuous backup).

  Hot continuous backup means that the OpenText eDirectory database is open and accessible during the process, and you still get a complete backup that is a snapshot of the moment when the backup began.
- Do a cold backup (the database is closed and a full backup is created).

  This option is helpful when upgrading hardware or moving a server to a new machine with the same operating system (as described in Upgrading Hardware or Replacing a Server).
- Set the database to stay closed and locked after a backup.
- Set the maximum backup file size.

Procedure

To back up the OpenText eDirectory database on a server using DSBK:

1. Enter the `dsbk backup` command, following this general pattern:

   `dsbk backup -b -f backup_filepath_and_backup_filename -l backup_log_filename_and_path -u include_file_filename_and_path -t -w`

   A space must be between each switch. The order of the switches is not important.

   For example, in Windows, enter the following command:

   ```
   dsbk backup -b -f c:\backups\8_20_2001.bak -l c:\backups\backup.log -u c:\backups\myincludefile.txt -t -w
   ```

   This example command would result in a full backup (`-b`) with the backup file placed at `c:\backups\8_20_2001.bak` and the log file for the process placed at `c:\backups\backup.log`. This command specifies that other files should be backed up along with the database:

   - The files listed in an include file (`-u c:\backups\myincludefile.txt`) that was created beforehand by the administrator.
   - Stream files (`-t`)

   This example command specifies that the backup file should be overwritten (`-w`), so if a file of the same name existed, the Backup Tool would replace it.

The output is entered in `dsbackup.out` file, which indicates whether the backup is successful.

Make sure you do a file system backup shortly after the OpenText eDirectory backup is created, to put the OpenText eDirectory backup files safely on tape. The Backup Tool only places them on the server.

Automating the Backing Up of OpenText eDirectory

To automate backing up of OpenText eDirectory, write the following command into a batch:

```
dhostcon.exe 192.168.1.1 load dsbk backup -b -f <Backup File> -l <Log File>
-t -w
```

For example,

```
c:\novell\nds\dhostcon.exe 192.168.1.1 load dsbk backup -b -f
edirbackup.bak -l c:\novell\edir-backup.log -t -w
```

Save this file in the location where you have installed eDirectory.

Configuring Roll-Forward Logs with DSBK

Use DSBK to change the settings for roll-forward logs. You can do the following tasks:

- Find out the current settings
- Turn roll-forward logging on or off

  You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open.
- Change the roll-forward logs directory
- Set the minimum and maximum roll-forward log size
- Find out the current and last unused roll-forward log
- Turn stream file logging on or off for the roll-forward logs

For information about roll-forward logging, see “Using Roll-Forward Logs” on page 432.

Procedure

1 Find out the current settings by entering

```
dsbk getconfig
```

No switches are necessary.

The following is an example of the information you receive:

```
Roll forward log status OFF
Stream file logging status OFF
Current roll forward log directory C:\rfl\nds.rfl
Minimum roll forward log size (bytes) 104857600
Maximum roll forward log size (bytes) 4294705152
Last roll forward log not used 00000000.log
Current roll forward log 00000001.log
*** END ***
```

2 Change the settings using the `setconfig` command, following this general pattern:

`dsbk setconfig [-L|-l] [-T|-t] -r path_to_roll-forward_logs -n minimum_file_size -m maximum_file_size`

A space must be between each switch. The order of the switches is not important.

Ideally, you would have a separate disk partition/volume dedicated to roll-forward logs to make it easier to monitor disk space and rights.

WARNING: If you turn on roll-forward logging, don't use the default location. For fault tolerance, put the directory on a different disk partition/volume and storage device than OpenText eDirectory. The roll-forward logs directory must be on the server where the backup configuration is being changed.

IMPORTANT: If you turn on roll-forward logging, you must monitor disk space on the volume where you place the roll-forward logs. If left unchecked, the log file directory will grow until it fills up the disk partition/volume. If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory stops responding on that server. We recommend you periodically back up and remove unused roll-forward logs from your server. See "Backing Up and Removing Roll-Forward Logs" on page 435.

Restoring from Backup Files with DSBK

Use DSBK to restore an OpenText eDirectory database from data stored in backup files you created manually. The results of the restore process are written to the log file you specify.

DSBK also lets you use advanced restore options. They are described in "Backup and Restore Command Line Options" on page 446, under restore and restadv.

Additional Prerequisites

- ❒ Make sure OpenText eDirectory is installed and running on the server you are restoring to.

  For example, if the restore is necessary because of a failed storage device, you need to do a new installation of OpenText eDirectory on the new storage device. If you are restoring a failed server onto a brand new machine, or simply moving a server from one machine to another, you need to install both the operating system and OpenText eDirectory on the new machine.
- ❒ Review the description of the command line options in "Backup and Restore Command Line Options" on page 446.
- ❒ Review the description of the restore process in "Overview of How the Backup Tool Does a Restore" on page 423.

Procedure

To restore an OpenText eDirectory database on a server using DSBK:

1 Make sure you have gathered the backup files you need, as described in "Preparing for a Restore" on page 436.

2 Enter the `dsbk restore` command, following this general pattern:

`dsbk restore -r -a -o -f full_backup_path_and_filename -d roll-forward_log_location -l restore_log_path_and_filename`

A space must be between each switch. The order of the switches is not important. Make sure you use the `-r` switch to restore the OpenText eDirectory database itself. Otherwise only the other kinds of files will be restored. If you want the database to be active and open when the restore is complete, make sure you specify `-a` and `-o`.

If you are restoring roll-forward logs, make sure you include the full path to the logs, including the directory that is automatically created by OpenText eDirectory, usually named `\nds.rfl`. For more information about this directory, see "Location of the Roll-Forward Logs" on page 434.

For example:

```
dsbk restore -r -a -o -f $HOME/backup/nds.bak -d $HOME/backup/rfldir/
nds.rfl -l $HOME/backup/backup.log
```

This example command specifies that the database itself should be restored (`-r`), and it should be activated (`-a`) and opened (`-o`) after the restore verification is successfully completed. The `-f` switch indicates where the full backup file is, `-d` the roll-forward logs, and `-l` the log file in which to record the results of the restore.

DSBK restores the full backup. The output is entered in `ndsd.log`, which will indicates whether the restore was successful.

3 (Conditional) If the restore was not successful, check the log file to see the errors.

If the restore verification fails, see "Recovering the Database If Restore Verification Fails" on page 456.

NOTE: If the server you are restoring shares a replica with a server running an earlier version than eDirectory 8.5, the restore log will show a -666 error (incompatible DS version) for that replica.

---

4 (Conditional) If you restored NICI security files, after completing the restore, restart the server to reinitialize NICI and then restore DIB.

5 Make sure the server is responding as usual.

6 (Conditional) If you are using roll-forward logging on this server, you must re-create your configuration for roll-forward logging to make sure it is turned on and the logs are being saved in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

This step is necessary because during a restore, the configuration for roll-forward logging is set back to the default, which means that roll-forward logging is turned off and the location is set back to the default. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

For more information about roll-forward logs and their location, see "Using Roll-Forward Logs" on page 432.

Your restore should now be complete, and NICI reinitialized with the restored NICI files so you can access encrypted information. If you use roll-forward logging, you have prepared for any failures in the future by turning on roll-forward logging again after the restore and creating a new full backup as a baseline.

Backup and Restore Command Line Options

The OpenText eDirectory Backup Tool command line options are divided into six functions: backup, restore, restadv, getconfig, setconfig, and cancel.

The switches can be placed in any order in the command after the name of the function. They must be separated by a space.

Option and Switches	Description
`backup`	Perform a backup of the database and associated files.
`-f` *`file_name`*	(Mandatory) Backup filename and path
	Specifies the filename and location of the backup file you want the Backup Tool to create. This file must be on the server you are backing up. For example, `backup -f C:\backup\ndsbak.bak` will back up the database to `C:\backup\ndsbak.bak`.
`-l` *`file_name`*	(Mandatory) Log filename and path
	Specifies the log file to record the results of the backup operation.
`-b`	(Optional) Perform a full backup.
	Performs a full backup of the OpenText eDirectory database. This option is the default behavior. If neither `-i` nor `-c` is specified, a full backup is performed.
`-i`	(Optional) Perform an incremental backup.
	Performs an incremental backup of the OpenText eDirectory database. This will back up any changes made to the database since the last full or incremental backup.
`-t`	(Optional) Back up stream files.
	Includes the stream files when backing up the OpenTexteDirectory database.
`-u` *`file_name`*	(Optional) User includes filename and path.
	Specifies an include file that lists additional files to back up. You can create this configuration file to include other files in the backup that could be important when restoring the server's OpenText eDirectory database.
	In the include file, list the full path of each file you want backed up, followed by a semicolon (;).
	Don't include any spaces or hard returns in the list of files.
	To confirm that these files are being backed up, check the backup log or look at the header of the backup file. See “Format of the Backup Log File” on page 429 and “Format of the Backup File Header” on page 424.
	WARNING: When opening a backup file, just view the header — make sure you don't try to save or modify the file, or it might become truncated. Most applications can't save the binary data correctly.

Option and Switches	Description
`-s file_size`	(Optional) Backup file size limit (MB) Specifies the maximum size (MB) of the backup file. You might want to use this option if you are concerned about file size because of the media you are using to store the backup files after they are created. If the maximum size is reached, a new backup file is created with the same name as the first with a five-digit hex extension added to denote what file it is. This extension increments with each new file. For example, you could set the maximum size of the backup files to 10 MB using the following switches as part of your command: `backup -f C:\backup\mydib.bak -s 10`. If the database is 35 MB, this is the resulting set of backup files: `C:\backup\mydib.bak`, size is 9.6 MB `C:\backup\mydib.bak.00001`, size is 9.6 MB `C:\backup\mydib.bak.00002`, size is 9.6 MB `C:\backup\mydib.bak.00003`, size is 5.6 MB The smallest possible size is close to 1 MB. The first file could be larger, depending on how many files are being included with the backup. The first file contains an attribute under the backup tag called `number_of_files`. This is the total number of files in the backup set. For the above example, this number would be 4. Also, the header of each backup file contains an attribute called `backup_file`. This is the original name of the file. For more information, see "Format of the Backup File Header" on page 424. When restoring a set of backup files like the set in the example above, the command would be `restore -f C:\backup\mydib.bak -l log_file_path_and_filename` The Backup Tool identifies that there are multiple files and looks for them in the same directory as the first, but with the above name mutations. **TIP:** The backup files can also be made much smaller using a third-party file compression tool. They compress approximately 80%.

Option and Switches	Description
`-w`	(Optional) Overwrite existing backup file of same name Overwrites the backup file specified with the `-f` switch if a file of the same name already exists. If this option is not used and a file of the same name already exists, in interactive mode the Backup Tool will ask you whether to overwrite or not. In batch mode, if a file of the same name exists and `-w` is not specified, the default behavior is to not overwrite the file, so a backup will not be created. If you are making a file system backup shortly after each full or incremental backup of OpenText eDirectory, your previous backup files should have been copied from the server to file system backup tapes, so it should be safe to use this option to overwrite the existing backup file. **IMPORTANT:** Use this option in your batch files for unattended backups. If a backup file of the same name exists (this is likely if you use the same batch file regularly), it's important to use the `-w` option to overwrite the existing backup file to make sure your backup is successful. In batch mode, if `-w` is not specified and a file of the same name exists, the default behavior is to not overwrite the file, so a backup will not be created. In interactive mode, if `-w` is not specified, DSBK will ask you whether you want to overwrite the file.
`-c`	(Optional) Perform a cold backup Performs a full backup of the database, but closes the database before the backup. After the backup has completed, the database reopens unless the `-o` or `-o` and `-d` switches are used.
`-o`	(Optional) Leave database closed after cold backup Can be used only if the `-c` switch is also used. Leaves the database closed after a cold backup. This option is helpful when upgrading hardware or moving a server to a new machine with the same operating system (as described in "Upgrading Hardware or Replacing a Server" on page 541).
`-d`	(Optional) Disable DS agent after a cold backup Can be used only if both the `-c` and `-o` switches are also used. Disables the DS agent after a cold backup. This option is helpful when upgrading hardware or moving a server to a new machine with the same operating system (as described in "Upgrading Hardware or Replacing a Server" on page 541). The DS agent is disabled by setting the login disabled attribute on the pseudo server. This results in a -663 error when OpenText eDirectory starts.
`-e` *`password`*	Perform a NICI backup *password* specifies the NICI backup password. This same password has to be specified to restore the NICI files.

Option and Switches	Description
`--config-file` *configuration file*	(Optional) Allows you to specify the instance of OpenText eDirectory you want to back up. *configuration file* specifies the absolute path to the configuration file of the eDirectory instance you want to back up. For example: `--config-file /etc/opt/novell/eDirectory/conf/nds.conf` This switch is applicable only for Linux environments.
`restore`	Perform a restore of the database and associated files.
`-f` *file_name*	(Mandatory) Backup filename and path Specifies which full backup to restore from. This file must be located on the server being restored. For example, `restore -f C:\backup\ndsbak.bak` will restore from the file `C:\backup\ndsbak.bak`. If the backup was made up of more than one file, all the files in the set must be copied into the same directory on the server.
`-l` *file_name*	(Mandatory) Log filename and path Specifies the log file to record the results of the restore operation.
`-r`	(Optional) Restore DIB set Specifies that the OpenText eDirectory database should be restored. **WARNING:** If you omit this option, the OpenText eDirectory database itself will not be restored. The only files that will be restored are other kinds of files you specify.
`-d` *dir_name*	(Optional) Roll-forward log directory Specifies the directory where the roll-forward logs are located. This must be the entire path and must be on the server being restored. All the roll-forward logs must be in the directory specified and they must have the same filenames as they did at the time of creation. After the database is restored, the changes recorded in these logs are replayed against the database to bring it up to date. If the `-d` switch is not used, the Backup Tool does not replay any logs against the database, even if roll-forward logging was turned on at the time of the backup. To determine the first required roll-forward log, open the last backup file being restored in a text editor and read the `current_log` attribute of the `backup` tag. The last backup file being restored is either the full backup file specified by the `-f` option or the last incremental backup file that is to be applied during the restore. For more information about the attributes listed in the header, see “Format of the Backup File Header” on page 424. **WARNING:** When opening a backup file, just view the header — make sure you don't try to save or modify the file, or it might become truncated. Most applications can't save the binary data correctly.

Option and Switches	Description
`-u`	(Optional) Restore user included files Restores the user files that were included with the backup of the database. As part of the backup, you can create a text file containing a list of files that you want backed up along with the database, and specify that file as the user includes file. These files will not be available to restore unless they were included in the backup.
`-a`	(Optional) Activate DIB after verifying Renames the database from RST to NDS after the restore verification completes successfully. For an overview of the process, see "Overview of How the Backup Tool Does a Restore" on page 423.
`-o`	(Optional) Open database when finished Directs the Backup Tool to open the database when the operation is complete. If the restore verification is successful, it opens the restored database. If the restore verification fails, this option opens the database that was on the machine before the restore was performed. For an overview of the process, see "Overview of How the Backup Tool Does a Restore" on page 423.
`-s`	Directs the Backup Tool not to reset roll forward log after Restore operation. It is mainly used in the instance of default RFL location.
`-n`	(Optional) Do not verify database after restore Directs the Backup Tool to restore the database without verifying. The transitive vector of this server will not be compared with the one expected by other servers in the replica ring it participates in. For information about transitive vectors, see "Transitive Vectors and the Restore Verification Process" on page 431. The database is not renamed from RST to NDS unless another option is used to do so. **IMPORTANT:** We do not recommend using this option unless suggested by OpenText Support.
`-v`	(Optional) Override restore Renames the database from RST to NDS without trying to verify. **IMPORTANT:** We do not recommend using this option unless suggested by OpenText Support.
`-k`	(Optional) Remove lockout on database Removes the lockout on the NDS database.
`-i`	Comma separated list of incremental files in order.
`-e` *`password`*	Restore the backed up NICI files *`password`* specifies the NICI backup password that was used when the NICI files were backed up. If a wrong password is specified when trying to restore the NICI files then an error message is displayed.

Option and Switches	Description
`--config-file` *`configuration file`*	(Optional) Allows you to specify the instance of OpenText eDirectory you want to restore. *configuration file* specifies the absolute path to the configuration file of the OpenText eDirectory instance you want to restore. For example: `--config-file /etc/opt/novell/eDirectory/conf/nds.conf` This switch is applicable only for Linux environments.
`restadv`	Advanced restore options. **NOTE:** The DS agent will be closed for all advanced restore options.
`-l` *`file_name`*	(Mandatory) Log filename and path Specifies the log file to record the results of the restore operation.
`-o`	(Optional) Open database when finished Directs the Backup Tool to open the database when the operation is complete. If the restore verification is successful, it opens the restored database. If the restore verification fails, this option opens the database that was on the machine before the restore was performed. For an overview of the process, see "Overview of How the Backup Tool Does a Restore" on page 423.
`-n`	(Optional) Try to verify a previously failed restore Tries to verify a previously restored RST database.
`-m`	(Optional) Remove restored DIB files Removes the RST database if it is present.
`-v`	(Optional) Override restore Renames the database from RST to NDS without trying to verify. **IMPORTANT:** We do not recommend using this option unless suggested by OpenText Support.
`-k`	(Optional) Remove lockout on database Removes the lockout on the NDS database.
`-i`	Comma separated list of incremental files in order. **IMPORTANT:** This option is applicable to DSBK only.
`getconfig`	Retrieves the current roll-forward log configuration.

Option and Switches	Description

	No options are needed.
	
	Displays the current settings. For example, on a server with roll-forward logging turned off, the `getconfig` command would return information like the following:
	
	`  Roll forward log status OFF`
	`Stream file logging status OFF`
	`Current roll forward log directory C:\rfl\nds.rfl`
	`Minimum roll forward log size (bytes) 104857600`
	`Maximum roll forward log size (bytes) 4294705152`
	`Last roll forward log not used 00000000.log`
	`Current roll forward log 00000001.log`
	`*** END ***`

`setconfig`	Sets the roll-forward log configuration.
`-L`	(Optional) Start keeping roll-forward logs.
	Turns on roll-forward logging. (Default=`Off`) Using continuous roll-forward logging lets you restore a server to the state it was in at the moment before it went down, instead of just to the last full or incremental backup. You must use roll-forward logging for servers that participate in replica ring, so that you can restore a server back to the synchronization state that the other servers expect. Administrative intervention is required after the roll-forward logs have been turned on. If left unchecked, the roll-forward logs continue to grow until they fill up the disk partition/volume. If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory stops responding on that server. Periodically, it is necessary to back up and delete unused logs. See “Backing Up and Removing Roll-Forward Logs” on page 435. For more information, see “Using Roll-Forward Logs” on page 432.
`-l`	(Optional) Stop keeping roll-forward logs
	Turns off roll-forward logging. (Default=`off`) The database reuses the current roll-forward log instead of saving a consecutive set of logs. If the roll-forward logs are turned off, you can restore OpenText eDirectory only to the point of the last full or incremental backup. If the logs are turned off unintentionally, you need to turn them back on and then do a new backup of the database to ensure that you can make a full recovery. For more information, see “Using Roll-Forward Logs” on page 432.

Option and Switches	Description
`-T`	(Optional) Start logging of stream files (Only applicable if the roll-forward logs are turned on.) Copies the entire stream file into the roll-forward log if a stream file is modified. Stream files are additional information files that are related to the database, such as login scripts. Roll-forward logs will fill disk space faster when stream files are being logged. Make sure you monitor disk space on the disk partition/volume where roll-forward logs are placed. If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory stops responding on that server.
`-t`	(Optional) Stop logging of stream files Stops copying the entire stream file into the roll-forward log if a stream file is modified. If roll-forward logging of stream files is turned off, you can use the backup options to back up stream files during full and incremental backup. Backing them up this way might be sufficient if your stream files don't change often. Turning off logging of stream files can help slow the growth of roll-forward logs.
`-r` *`dir_name`*	(Optional) Set roll-forward log directory Changes the directory where the roll-forward logs are placed. For example, if the command used was `setconfig -r vol2:\rfl`, a directory is created under `vol2:\rfl` and the roll-forward logs are placed in it. This directory name is based on the name of the current OpenText eDirectory database. For typical installs this is NDS, so the final directory name would be `vol2:\rfl\nds.rfl\`. If you renamed the OpenText eDirectory database from NDS to ND1, the roll-forward log directory would be changed to `vol2:\rfl\nd1.rfl\`. You can find out the current location by entering the `getconfig` command. When you change the location, the new directory is created immediately, but a roll-forward log is not created there until a transaction takes place in the database. **IMPORTANT:** The Backup tool has no way of tracking the changes to the roll-forward log directory. When restoring the database, you must collect all roll-forward logs and place them in one directory on the server. For more information, see "Using Roll-Forward Logs" on page 432.
`-n` *`file_size`*	(Optional) Set minimum roll-forward log size Sets the minimum size of the roll-forward log files (in bytes). When the minimum size is reached, the database starts a new roll-forward log after the current transaction is finished.

Option and Switches	Description
`-m file_size`	(Optional) Set maximum roll-forward log size Sets the maximum size for the roll-forward log files (in bytes). If this limit is reached and a transaction is in progress, the transaction is continued over into the next file. This setting must always be larger than the minimum size.
`-s`	(Optional) Start a new roll-forward log Starts a new roll-forward log at the end of the current transaction. The new file is created at the beginning of the next transaction.
`cancel`	Cancels any running backup or restore operation. No options are needed. **NOTE:** This option is not applicable to DSBK.
`--config-file configuration file`	(Optional) Allows you to specify the instance of OpenText eDirectory for which you want to set the roll-forward log configuration. *configuration file* specifies the absolute path to the configuration file of the OpenText eDirectory instance for which you want to set the roll-forward log configuration. For example: `--config-file /etc/opt/novell/eDirectory/conf/nds.conf` This switch is applicable only for Linux environments.

Running DSBK as a cron Job

The `dsbk` script does not contain the full path to the DSTrace binary. Therefore, if you run the script as a cron job using the default settings, the script fails. However, do not change the `/opt/novell/eDirectory/bin/dsbk` script to add the path, because subsequent OpenText eDirectory patches will overwrite this file and revert any customizations you may have made to the script.

Instead, before you run dsbk as a cron job, set the PATH environment variable within the `crontab` file to include the directory where `ndstrace` is located. The cron job can then find and run the `ndstrace` application.

Backing Up and Restoring NICI

NICI stores keys and user data in the file system and in system and user specific directories and files. These directories and files are protected by setting the proper permissions on them using the mechanism provided by the operating system. This is done by the NICI installation program. NICI back up and restore is supported only for a root user, and not for a non-root user.

Uninstalling NICI from the system does not remove the system or user directories and files. Therefore, the only reason to restore these files to a previous state is to recover from a catastrophic system failure or a human error. It is important to understand that overwriting an existing set of NICI user directories and files might break an existing application.

The database key required to open the DIB is wrapped with NICI keys. Hence if an OpenText eDirectory backup is performed independent of NICI backup then it is of no use. The OpenText eDirectory backup solution (DSBK and eMBox Backup) has a switch (`-e`) that enables:

1. Backing up the NICI keys when an OpenText eDirectory backup is run
2. Restoring the NICI keys when an OpenText eDirectory restore is run

For more information on the OpenText eDirectory backup solution, refer to the "Using DSBK" on page 439.

Backing Up NICI

NICI backup can be performed along with full OpenText eDirectory backup and also with incremental OpenText eDirectory backup.

The command to perform a NICI backup is as follows:

```
dsbk backup -f file_name -l log_file_name -e password
```

`-f` and `-l` are mandatory options that have to be used with the backup command.

`-e` is the switch to backup NICI files.

`file_name` specifies the file name and location of the backup file you want the Backup Tool to create.

`log_file_name` specifies the file name and location of a log file created to record the results of the backup operation.

`password` specifies the NICI backup password. The password can be specified as a clear text. On Linux, passing the password as a file is also supported. This same password has to be specified to restore the NICI files.

NOTE: If a NICI backup password is not specified with the `-e` switch, then the following error messages are displayed:

In DSBK:

```
Enter password along with the (-e) option!
DSBK error! 4
```

Restoring NICI

1 Restore NICI files alone (not DIB).

```
dsbk restore -f file_name -l log_file_name -e password
```

`-f` and `-l` are mandatory options that have to be used with the restore command.

`-e` is the switch to restore NICI files.

`file_name` specifies the file name and location of the backup file that contains the information to be restored.`log_file_name` specifies the file name and location of a log file created to record the results of the restore operation.`password` specifies the NICI backup password that was used when the NICI files were backed up. If a wrong password is specified when trying to restore the NICI files then an error message is displayed.

2 Restart the ndsd server.

3 Restore the DIB.

```
dsbk restore -f file_name -l log_file_name -a -r -o
```

`-f` and `-l` are mandatory options that have to be used with the restore command.

`-a` activates DIB after verifying, `-r` restores DIB set, and `-o` opens database when finished.

If NICI backup was performed during a full backup and also during an incremental backup and if different NICI backup passwords were used during the full backup and the incremental backup then when restoring the NICI files the password that was used with the full backup should be used to restore the NICI files.

NOTE: If a password is not specified with the `-e` switch then the following error messages are displayed:

In DSBK:

```
Enter password along with the (-e) option!
DSBK error! 4
```

If a wrong password is specified during the NICI restore, the following error is displayed:

```
NICI RESTORE: "NICI Files has not been restored(Check your parameters)"
Error!: -32
```

Recovering the Database If Restore Verification Fails

The restore process includes a verification step, which compares the OpenText eDirectory database on the server being restored to other servers in the replica ring by comparing the transitive vectors. For more information on the restore process, see “Overview of How the Backup Tool Does a Restore” on page 423 and “Transitive Vectors and the Restore Verification Process” on page 431.

If the transitive vectors do not match, the verification fails. This usually indicates that data is missing from the files you used for the restore. For example, data might be missing for the following reasons:

- You did not turn on roll-forward logging before the last backup was performed.
- You did not include the roll-forward logs in the restore.
- The set of roll-forward logs you provided for the restore was not complete.

By default, the restored OpenText eDirectory database will not open after the restore if it is inconsistent with the other replicas.

If you have all the backup files and roll-forward logs necessary for a complete restore but forgot to provide all of them during the process, you can simply run the restore again with a complete set of files. If the restore is complete on a second try, the verification can succeed and the restored database will open.

If you do not have all the backup files and roll-forward logs necessary to make the restore complete so that verification will be successful, you must follow the instructions in this section to recover the server. Here is an outline of what you can recover if verification fails:

- You can still recover the server's identity and file system rights.

- You cannot recover any replicas on this server from backup, but the server can still be used for the replicas it contained after you follow the recovery procedure in this section. You must remove the server from the replica ring and use advanced Restore options and the DSRepair Tool to bring the server to a state where it can be put back in the replica ring. Then you can re-add the desired replicas to it.
- Unfortunately, if this server had the sole copy of any partition of the database (there were no other replicas of the partition), the partition cannot be recovered.

Use the instructions in this section after verification fails to recover the server's identity and file system rights, and to remove and re-add it to the replica ring. When you have followed these steps and the replication process is complete, the server should function as it did before the failure (with the exception of any partitions that were not replicated and, therefore, can't be recovered).

First, complete "Cleaning Up the Replica Ring" on page 457. Then continue with "Repair the Failed Server and Re-add Replicas to the Server" on page 458.

Cleaning Up the Replica Ring

This procedure explains how to,

- **Reassign master replicas.** If the failed server holds a master replica of any partition, you must use DSRepair to designate a new master replica on a different server in the replica list.
- **Remove replica list references to the failed server.** Each server participating in replica ring that included the failed server must be told that the failed server is no longer available.

Prerequisites

- ❒ OpenText eDirectory is installed on the machine where you are trying to restore the failed server.
- ❒ A restore was attempted, and the restore verification failed.
- ❒ The OpenText eDirectory database is open and running, and the database named RST is still on the machine (left there by the restore process).
- ❒ You know which replicated partitions were stored on the failed server. The replicas this server held are listed in the header of the backup file.

Procedure

To clean up the replica ring:

1 At the console of one of the servers that shared a replica with the failed server, load DSRepair with the switch that lets you access the advanced options.

- **Windows:** Use the `-a` switch.
- **Linux:** Use the `-Ad` switch.

For more information on how to run DSRepair with advanced options using the `-a` or `-Ad` switches, see "DSRepair Options" on page 328.

WARNING: If you use DSRepair with `-a` or `-Ad`, some of the advanced options can cause damage to your tree.

---

2 Select **Replica and Partition Operations**.

3 Select the partition you want to edit, so you can remove the failed server from the replica ring of that partition.

4 Select **View Replica Ring** to see a list of servers that have replicas of the partition.

5 (Conditional) If the failed server held the master replica, select another server to hold the master by selecting **Designate This Server As the New Master Replica**.

The replica ring now has a new master replica. All replicas participating in the ring are notified that there is a new master.

6 Wait for the master replica to be established. Make sure the other servers in the ring acknowledge the change before proceeding.

7 Go back to **View Replica Ring**. Select the name of the failed server, then select **Remove This Server from the Replica Ring**.

If you have not loaded DSRepair with `-a` or `-Ad` (depending on the platform) for advanced options, you will not see this option in the list.

WARNING: Make sure you do not do this if the failed server is designated as the master replica. You can see this information in the list of servers in the ring. If it is the master, designate a different server as the master as noted in Step 5. Then, come back to this step and remove the failed server from the replica ring.

---

8 Log in as Admin.

9 After reading the explanation message, enter your agreement to continue.

10 Exit DSRepair.

All servers participating in that replica ring are notified.

11 Repeat this procedure on one server for each replica ring that the failed server participated in.

To finish preparing the failed server to get new copies of the replicas, continue with the next procedure, "Repair the Failed Server and Re-add Replicas to the Server" on page 458.

Repair the Failed Server and Re-add Replicas to the Server

This procedure lets you change the replica information on the server to external references, so that the server does not consider itself to be part of a replica ring. After you remove the replicas from the server in this way, you can unlock the database.

After removing the replicas, you complete the procedure by re-adding the replicas to the server. This way, the server receives a new, up-to-date copy of each replica. When each replica has been re-added, the server should function as it did before the failure.

To remove replicas using DSRepair, and re-add them using replication:

1 Make sure you have completed "Cleaning Up the Replica Ring" on page 457.

2 Specify the advanced restore option to override the restore, then specify a log filename:

```
dsbk restadv -v -l logfilename
```

This advanced restore option renames the RST database (the database that was restored but failed the verification) to NDS, but keep the database locked.

3 At the server console, change all the replica information on the server into external references using advanced options in DSRepair.

- **Windows:** Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**. Select **dsrepair.dlm**. In the Startup Parameters field, type `-XK2 -rd`. Click **Start**.
- **Linux:** Enter the following command:

  ```
  ndsrepair -P -Ad
  ```

The `-rd` or `-P` switch repairs the local database and the replica.

WARNING: If used incorrectly, DSRepair advanced options can cause damage to your tree.

---

4 When the repair is finished, remove the lockout and open the database using the following advanced restore options in the eMBox Client:

```
dsbk restadv -o -k -l logfilename
```

The `-o` opens the database and the `-k` removes the lockout.

5 Use OpenText Identity Console to add the server back into the replica ring:

5a On the OpenText Identity Console home page, click the **Partition Management** tile.

5b On the **Partitions** page, click **Type** drop down menu > select **Server** > click **Search**.

5c Select the partition you want to replicate.

5d On the **Replica View** page > click **Add Replica** +.

5e Click **Partition name**, then select the server you just restored.

5f Select the type of replica you want, click **OK**, then click **Done**.

5g Repeat these steps for each replica ring that the server was participating in.

6 Wait for the replication process to complete.

The replication process is complete when the state of the replicas changes from New to On. You can check the state in OpenText Identity Console. See "Viewing Information about a Replica" on page 154 for more information.

7 To restore NICI security files, first restore the NICI files alone and then restart the NDSD server and restore the DIB.

8 (Conditional) If you want to use roll-forward logging on this server, you must re-create your configuration for roll-forward logging to make sure it is turned on and the logs are being saved in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

This step is necessary because during a restore, the configuration for roll-forward logging is set back to the default, which means that roll-forward logging is turned off and the location is set back to the default. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

For more information about roll-forward logs and their location, see "Using Roll-Forward Logs" on page 432.

Scenarios for Backup and Restore

- “Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Single-Server NetWork” on page 460
- “Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Multiserver Environment” on page 461
- “Scenario: Losing an Entire Server in a Multiple-Server Environment” on page 463
- “Scenario: Losing Some Servers in a Multiple-Server Environment” on page 464
- “Scenario: Losing All Servers in a Multiple-Server Environment” on page 464

Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Single-Server NetWork

Indira is the administrator for a single-server network at Stationery Supply, Inc. Indira can't rely on replication for fault tolerance, because her environment has only one server. The Backup Tool functionality provides a simple solution for Indira to back up and restore OpenText eDirectory. It's server-centric and it's fast.

On eDirectory 8.7.3 or to later versions, Indira sets up unattended backups for her server using batch files to run the Backup Tool.

Indira wants to do a full backup of OpenText eDirectory every Sunday night, and an incremental backup every weeknight. She sets the unattended backups to run shortly before her full and incremental file system backups each night, so her tape backups contain the OpenText eDirectory backup files as well as the file system data. She has contracted with a remote data storage company to send the tape backups offsite.

Every Monday morning, Indira checks the backup log to make sure the full backup was successful. She also checks the logs occasionally during the week to make sure the incremental backups were successful.

Indira decides not to turn on roll-forward logs for the following reasons:

- She does not have a separate storage device on her server, so turning on roll-forward logs would not provide any additional backup of OpenText eDirectory. If there were a storage device failure, the logs would be lost along with OpenText eDirectory, so there is no point in creating them.
- The tree does not change very much, and she is satisfied with being able to restore only up to last night's backup. She doesn’t need to be able to restore OpenText eDirectory to the moment before a failure.
- Because the server does not participate in a replica ring with other servers, roll-forward logs are not required for the restore verification process to be successful.

Stationery Supply, Inc. decides to reorganize the staff, so Indira does a manual backup before and after making significant changes to the tree. Her strategy is to make a new backup of changes during the middle of a weekday when necessary, instead of running roll-forward logs all the time.

To make sure her backup strategy is ready to go when she needs it, Indira tests it occasionally. She doesn’t have the budget to purchase a second server for testing, so she makes arrangements with a test lab in her town. Using a server like hers in the test lab, she installs her operating system and tries to approximate the environment of her OpenText eDirectory database. She restores her backups and checks to make sure OpenText eDirectory is restored as she expects.

One Wednesday morning, the hard drive containing OpenText eDirectory on the server has a failure. Indira obtains a new hard drive and the backup files from the full backup on Sunday evening, the incremental backup on Monday evening, and the incremental backup on Tuesday evening. She installs the new hard drive and installs OpenText eDirectory on it. Then she restores the full and incremental backups. Any changes to the tree that were made on Wednesday morning before the hard drive failure are lost because Indira was not running roll-forward logs on the server. But Indira is satisfied with restoring only to last night's backup. She doesn’t feel that running roll-forward logs would be worth the administrative overhead.

Scenario: Losing a Hard Drive Containing OpenText eDirectory in a Multiserver Environment

Jorge at Outdoor Recreation, Inc. has 10 servers running OpenText eDirectory. He does full backups every Sunday night and incremental backups nightly, running the OpenText eDirectory backup shortly before the file system backup to tape.

All of the servers are participating in replica ring. Jorge uses roll-forward logging for all the servers. On each of his servers, he has placed the roll-forward logs on a different storage device than OpenText eDirectory. He monitors the free space and rights on those storage devices to make sure the roll-forward logs don't fill up the storage device. Occasionally he backs up the roll-forward logs to tape and removes all except the one in use by OpenText eDirectory, to free up space.

The administrative overhead of turning on continuous roll-forward logging is worth it to Jorge, because it gives him the up-to-the moment backup required for servers that participate in replica ring. This way, if he needs to restore a server, the restored server will match the synchronization state that other servers in the replica ring expect.

In his test lab, Jorge periodically tests his backup files to make sure his backup strategy will meet his goals.

One Thursday at 2:00 p.m., the Linux server named Inventory_DB1 has a hard drive failure on the drive containing OpenText eDirectory.

Jorge needs to gather the last full backup and the incremental backups since then, which will restore the database up to the point of last night's incremental backup at 1:00 a.m. The roll-forward logs have been recording the changes to the database since last night's backup, so Jorge will include them in the restore to bring the database back to the state it was in just before the hard drive failure.

Jorge takes the following steps:

1. He gets a replacement hard drive for the server.
2. He gets the tape of the full backup for the server from the previous Sunday night.

   The batch file he uses to run full backups every Sunday night places the backup file in `/adminfiles/backup/backupfull.bk`.

   He had specified a file size limit of 200 MB in the backup configuration settings, so there are two backup files:

`backupfull.bk.00001` (250 MB)

`backupfull.bk.00002` (32 MB)

3. He also gets the tapes containing the incremental backups for Monday, Tuesday, and Wednesday nights.

   The batch file he uses to run incremental backups every weeknight places the backup file in `/adminfiles/backup/backupincr.bk`.

   Because he runs the same batch file every weeknight for the incremental backups of OpenText eDirectory, they all have the same filename. He needs to give them new names when he copies them back onto the server, because they all must be placed in the same directory during the restore.

4. Jorge installs the replacement hard drive.

   In this case, the Linux operating system for the server was not on the hard drive that failed, so he does not need to install Linux.

5. Jorge restores the file system from tape backup for the disk partitions that were affected.
6. Jorge reinstalls eDirectory, putting the server into a new temporary tree (the restore puts it back into the original tree again later).
7. Jorge creates an `/adminfiles/restore` directory on the server, to hold the files to be restored.
8. He copies the full backup (the set of two files) into that directory.
9. He copies the incremental backups for Monday, Tuesday, and Wednesday nights into the directory.

   Each of them is named `backupincr.bk`, so when he copies them into the directory he changes the filenames to

   `backupincr.mon.bk`
   `backupincr.tues.bk`
   `backupincr.wed.bk`

   NOTE: Full and incremental backups aren't required to be in the same directory together, but all the incremental backups must be in the same directory.

   ---

10. He uses EmBox to restore OpenText eDirectory:
    a. He login to the server with the command: `edirutil -i`.
    b. He runs the command: `login -s 10.71.39.10 -u admin.novell -p 8028 -w novell -n`.
    c. He runs the command: `restore -r -a -o -f /embox/backup5000.bak -d /var/opt/novell/eDirectory/data/dib/nds.rfl -l /embox/backupNew.log`.
    d. In the Restore Wizard - Optional screen, he does the following:

       Checks **Restore Database**.

       Checks **Restore Roll-Forward Logs**.

       Enters the location of the roll-forward logs.

       (This is the separate location that he created specifically to hold the roll-forward logs. Because he placed them on a different hard drive than OpenText eDirectory, the hard drive failure did not affect them and they are still available.)

Checks **Restore Security Files**

Checks **Activate the Restored Database after Verification**.

Checks **Open the Database after Completion of Restore**.

Wants OpenText eDirectory to open if the restore verification is successful.

11. He starts the restore and enters the filenames of the incremental backup files when prompted.
12. The restore verification is successful, so the database opens, back in its original tree.

    The restore verification was successful because roll-forward logs were running on the server when the hard drive failed, and Jorge included the logs in the restore.
13. Jorge re-creates the roll-forward logs configuration on the server after the restore is complete, then he creates a new full backup.

    The settings are reset to the default during a restore, which means roll-forward logging is turned off, so he has to turn it back on. The new full backup is necessary so that he is prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

Jorge checks the way the server is running, and it appears to be normal.

Scenario: Losing an Entire Server in a Multiple-Server Environment

Bob is the administrator for 15 servers at GK Designs Company. He does full backups every Saturday night and incremental backups nightly, running the OpenText eDirectory backup shortly before the file system backup to tape.

All of the servers are participating in replica ring. Bob uses roll-forward logging for all the servers.

An electrical fire destroys one of the servers in a branch across town. Fortunately, all but one of the partitions held by this server are also replicated on other servers. Bob had turned on roll-forward logs on that server, but they were lost along with all the other server data, so he can't restore the OpenText eDirectory database on that server to the state it was in just before the server went down.

However, he is able re-create the server's OpenText eDirectory identity by restoring with the existing backup files. Because Bob can't include the roll-forward logs in the restore, the server does not match the synchronization state that the other servers expect (see “Transitive Vectors and the Restore Verification Process” on page 431), so the restore verification process is not successful. This means that by default the eDirectory database is not opened after the restore.

Bob addresses the situation by removing this server from the replica ring, using DSRepair to change all the outdated replica information on the server to external references, and then re-adding a new copy of each partition to this server using replication from the other servers that hold the up-to-date replicas. These steps are described in “Recovering the Database If Restore Verification Fails” on page 456.

The one partition on this server that Bob had not replicated was a container that held network printing objects for the branch office location, such as a fax/printer and a wide-format color printer. This partition information can't be recovered by the method noted above because no other server has a replica. Bob must re-create the objects in that partition, and this time he chooses to replicate them on other servers for better fault tolerance in the future.

Bob also re-creates the roll-forward log configuration after the server is back on line (because the restore turns it off and resets the settings to the default), and creates a new full backup as a baseline.

Scenario: Losing Some Servers in a Multiple-Server Environment

Joe administers 20 servers across three locations. At one location, a pipe bursts and water destroys 5 out of 8 servers.

Joe has OpenText eDirectory backups for all the servers. However, all the servers participate in replica ring, and he is concerned about bringing them back into the tree without the roll-forward logs, which were also lost. He is not sure which servers to restore OpenText eDirectory on first or how to address inconsistencies between replicas. Because of the complex issues involved, he calls OpenText Support for help in deciding how to restore.

Scenario: Losing All Servers in a Multiple-Server Environment

Delores and her team at Human Resources Consulting, Inc. administer 50 servers at one location.

For fault tolerance during normal business circumstances, they have created three replicas of each partition of their tree, so that if one server is down, the objects in the partitions it holds are still available from another server. They have also planned for recovery of individual servers by backing up all their servers regularly with the Backup Tool, turning on roll-forward logging, and storing the backup tapes at a remote location.

For disaster recovery planning, Delores and her team have also designated two of their servers as DSMASTER servers. They use two servers because their tree is large enough that more than one DSMASTER server is needed to hold a replica of every partition. Every partition in the tree is replicated on one of the two DSMASTER servers. Neither of the two DSMASTER servers hold replicas of the same partition, so there is no overlap between them. This design is an important part of their disaster recovery plan.

In their test lab, Delores and her team periodically test the backups to make sure their backup strategy will meet their goals.

One night the Human Resources Consulting, Inc. building is damaged by a hurricane, and all the servers in the data center are destroyed.

After this disaster, Delores and her team first restore the two DSMASTER servers, which hold replicas of every partition. They use the last full backup and the subsequent incremental backups, but can't include roll-forward logs in the restore because they were lost when the servers were destroyed. Delores and her team planned the DSMASTER servers so that they don't share replicas. Because the two DSMASTER servers do not share replicas, the restore verification process is successful for both servers even though the roll-forward logs are not part of the restore. After the DSMASTER servers are restored, all the objects in the tree for Human Resources Consulting, Inc. are now available again.

The DSMASTER servers are important because Delores and her team can use them to re-create the tree without inconsistencies after a disaster.

They were using roll-forward logs so they could restore a server to the state it was in at the moment before it went down, bringing it back to the synchronization state expected by other servers in the replica ring. This allows the server to resume communication where it left off, and receive any updates it needs from the other replicas to keep the whole replica ring in sync.

However, in this disaster situation, Delores and her team do not have the roll-forward logs. Without the roll-forward logs, only one server in a replica ring can be restored without errors—the first one they restore. For the rest of the servers, the restore verification process will fail because the synchronization states don't match what the other servers expect (see “Transitive Vectors and the Restore Verification Process” on page 431). If the restore verification fails, the restore process will not activate the restored OpenText eDirectory database.

Delores and her team anticipated this, and they have planned for it. They use the two DSMASTER servers as a starting point, which gives them only one replica of each partition.Those servers can be restored without verification errors, and then the replicas they hold can be used as masters to be copied onto all the other servers.

After restoring the DSMASTER servers, restoring the rest of the servers requires some extra steps. Delores and her team must restore each of the remaining servers by doing the following:

- Making sure that the replicas on the DSMASTER servers are designated as master replicas.
- Removing all the servers except the DSMASTER servers from the replica ring.
- Restoring the full and incremental backups for each of the other servers.

  Delores and her team know that the restore verification process will fail for the rest of the servers, because they could not use roll-forward logs in the restore for any of the servers. This leaves them with a restored database that is not activated.
- Activating the restored database, but keeping it locked, using advanced restore options
- Using DSREPAIR to change all the replica information to external references.
- Unlocking the restored database.

  At this point the server has the same identity it did before but it will not try to synchronize replica information. Instead, it is prepared to receive a new copy of the replicas it held before.
- Adding the replicas back on to each server by replicating them from the copy on the DSMASTER server.

  Delores and her team have a pretty good idea which replicas were held by each server, but they can read the header of the backup files for each server to see a list of the replicas that were on the server at the time of the last backup.
- Re-creating the roll-forward log configuration after the servers are back on line (since the restore turns it off and resets the settings to the default), and creating a new full backup as a baseline to prepare for any other failures that might happen before the next unattended full backup is scheduled.

(These steps are explained in more detail in “Recovering the Database If Restore Verification Fails” on page 456.)

Delores and her team have a lot of work to do, but they can get the tree itself up relatively quickly, and they can expect to recover the OpenText eDirectory identity for all of their servers.

Disaster Recovery Plan using DSBK

A disaster recovery plan enables you to recover your disk back to the configuration at the time of corruption. You have to backup your server's disk to a remote location, so that you can recover the server even if the operating system gets corrupted.

This section provides a sample disaster recovery plan for an OpenText eDirectory server:

- "Disaster Recovery Plan on Linux" on page 466
- "Disaster Recovery Plan on Windows" on page 467

Disaster Recovery Plan on Linux

To take a backup of the server's disk:

1. Configure DSBK:
   - **1a** Create a file `dsbk.conf` in `/etc`.
   - **1b** Create a temporary file. For example, `/tmp/dsbk.tmp`.
   - **1c** Specify the location of the temporary file created in the previous step in the `/etc/dsbk.conf` file.
2. Mount the server's disk to a remote machine in the read/write mode, to store all backup files on a remote machine disk.

   For example, `eDirServer# mount <remote machine IP>:/home/backup/ /mnt/dsbkBkp`
3. Set the custom backup location using the following command:

   ```
   dsbk setconfig -L -T -r /mnt/dsbkBkp
   ```

   NOTE: Ensure that you run DSBK from the following location on the server: `/opt/novell/eDirectory/bin`.
4. Take a full backup along with NICI to the remote location file system:

   ```
   dsbk backup -f <backup file location> -l <log file location> -e
   <password for NICI backup> -t -b
   ```

   For example, `dsbk backup -f /mnt/dsbkBkp/fb1.bak -l /mnt/dsbkBkp/fb1.log -e novell -t -b`.

   NOTE: The `-e` option is used to back up NICI. In the example, `novell` is password for NICI Backup. You may choose your own password, and the same password must be used during NICI restore.
5. Take incremental backups using the following command:

   ```
   dsbk backup -f <incremental backup file location> -l <incremental log
   file location> -t -i
   ```

   For example:

   Day 1: `dsbk backup -f /mnt/dsbkBkp/ib1.bak -l /mnt/dsbkBkp/ib1.log -t -i`

Day 2: `dsbk backup -f /mnt/dsbkBkp/ib2.bak -l /mnt/dsbkBkp/ib2.log -t -i`

NOTE: While taking an incremental backup, you do not have to back up NICI.

If the OpenText eDirectory server gets corrupted, then perform the following steps to recover the OpenText eDirectory server using the remote location backup:

1. If the operating system is corrupted, install the operating system as before.
2. If only OpenText eDirectory is corrupted, then do a clean up of the system for eDirectory by removing the OpenText eDirectory RPMs.
3. Install the same OpenText eDirectory as before and configure a single server dummy tree. For example,

   `ndsconfig new -t dummy_bkp_tree -n novell -a admin.novell -w novell`
4. Restore NICI from the full backup file (without the `-d`, `-r`, `-a`, `-o` options):

   `dsbk restore -f <backup file location> -l <log file location> -e <password used to NICI backup>`

   For example, `dsbk restore -f /mnt/dsbkBkp/fb1.bak -l /mnt/dsbkBkp/restore1.log -e novell`
5. After restoring NICI, restart the OpenText eDirectory server.
6. Restore both the full and incremental backup files. For example,

   `dsbk restore -f /mnt/dsbkBkp/fb1.bak -l /mnt/dsbkBkp/restore2.log -d /mnt/dsbkBkp/nds.rfl/ -r -a -e novell -o -i /mnt/dsbkBkp/ib1.bak, /mnt/dsbkBkp/ib2.bak`

For more information on backup and restore commands, refer to the "Using DSBK on Linux" on page 440.

Disaster Recovery Plan on Windows

To take a backup of the server's disk:

1. Map the server's disk to a remote machine in the read/write mode. For example, `O:\dsbkBkp`
2. To run DSBK command:
   - **2a** Open OpenText eDirectory server console by running `NDScons.exe`.
   - **2b** Click **dsbk.dlm** from the **Services** tab.
   - **2c** Enter DSBK commands in the **Startup Parameter** field.
3. Set the custom backup location using the following command:

   `setconfig -L -T -r O:\dsbkBkp`
4. Take a full backup along with NICI, to the remote location file system:

   `backup -f <backup file location> -l <log file location> -e <password for NICI backup> -t -b`

NOTE: The `-e` option is used to backup NICI. In the example, `novell` is password for NICI Backup. You may choose your own password, and the same password must be used during NICI restore.

5 Take incremental backups using the following command:

```
backup -f <incremental backup file location> -l <incremental log file location> -t -i
```

For example:

Day 1: `backup -f O:\dsbkBkp\ib1.bak -l O:\dsbkBkp\ib1.log -t -i`

Day 2: `backup -f O:\dsbkBkp\ib2.bak -l O:\dsbkBkp\ib2.log -t -i`

NOTE: While taking an incremental backup, you do not have to back up NICI.

If the OpenText eDirectory server gets corrupted, then perform the following steps to recover the OpenText eDirectory server using the remote location backup:

1 If the operating system is corrupted, install the operating system as before.

2 If only OpenText eDirectory is corrupted, then do a clean up of the system for OpenText eDirectory.

3 Install the same OpenText eDirectory as before and configure a single server dummy tree.

4 Restore NICI from the full backup file (without the `-d`, `-r`, `-a`, `-o` options):

For example:

```
restore -f <backup file location> -l <log file location> -e <password used for NICI backup>
```

For example, `restore -f O:\dsbkBkp\fb1.bak -l O:\dsbkBkp restore1.log -e novell`

5 After restoring NICI, restart the OpenText eDirectory server.

6 Restore both the full and incremental backup files.

For example:

```
restore -f O:\dsbkBkp\fb1.bak -l O:\dsbkBkp\restore2.log -d O:\dsbkBkp\nds.rfl\ -r -a -e novell -o -i O:\dsbkBkp\ib1.bak, O:\dsbkBkp\ib2.bak
```

For more information on backup and restore commands, refer to the "Using DSBK on Windows" on page 441.

LDAP-Based Backup

The LDAP-based backup feature is used to backup the attributes and attribute values one object at a time.

The following table lists the platforms that support this feature:

Feature	Linux	Windows
LDAP-based backup	✓	✓

This feature lets you perform an incremental backup wherein the object is backed up only if there are changes to the object.

LDAP-based backup provides a set of interfaces for backup and restore of OpenText eDirectory objects exposed through the LDAP Libraries for C through LDAP extended operations.

For more information on LDAP Libraries for C SDK, refer to the LDAP Libraries for C documentation (https://www.microfocus.com/documentation/edirectory-developer-documentation/ldap-libraries-for-c/#:~:text=LDAP%20Libraries%20for%20C%20enables,draft%20proposed%20to%20the%20IETF).

For an example of how to do backup and restore of OpenText eDirectory objects through LDAP, refer to the backup.c sample code (https://www.novell.com/documentation/developer/samplecode/cldap_sample/cldap_sample/extensions/backup.c.html).

Need for LDAP Based Backup

The LDAP based backup tries to resolve the problems with the current backup and restore.

The problems that this feature resolves are:

- Gives a consistent interface using which any third party backup applications or developers can backup OpenText eDirectory on all the supported platforms.
- Provides a backup solution to backup objects incrementally.

For More Information

For more information on this feature, refer to the following:

- LDAP Libraries for C (https://www.novell.com/documentation/developer/cldap/ldaplibc/data/a2etgcm.html)
- Sample code: backup.c (https://www.novell.com/documentation/developer/samplecode/cldap_sample/cldap_sample/extensions/backup.c.html)

OpenText eDirectory Backup with SMS

Storage Management Services (SMS) is an API framework consumed by backup applications to provide a complete backup solution. The SMS framework is implemented by two main components:

- Storage Management Data Requester (SMDR)
- Target Service Agent (TSA)

The TSA for the OpenText eDirectory (`tsands`) services OpenText eDirectory targets and provides an implementation of the Storage Management Services API for the directory trees. Applications can be written on top of `SMS API` to provide a complete backup solution.

NOTE: The TSA for OpenText eDirectory is only supported in OES environment.

---

16 Configuring OpenText eDirectory in Suite B Mode

Suite B is a set of cryptographic algorithms standardized by the National Security Agency (NSA) to allow commercial products to protect traffic that is classified at secret or top secret levels. The Suite B algorithms serve as a method to ensure the security of classified and unclassified information passed through public networks.

NOTE: Suite B standard is subject to change, be aware that NSA may change their recommendations in future. Suite B support in OpenText eDirectory is based on our interpretation of the NSA recommendations.

Suite B includes the following cryptographic algorithms:

- Encryption based on the Advanced Encryption Standard (AES) using 128-bit keys or 256-bit keys
- Digital signatures with the Elliptic Curve Digital Signature Algorithm (ECDSA) on P-256 and P-384 curves
- Key exchange, either pre-shared or dynamic, using the Elliptic Curve Diffie-Hellman (ECDH) method on P-256 and P-384 curves
- Hashing (digital fingerprinting) based on the Secure Hash Algorithm-2 (SHA-256 and SHA-384)

For more information about Suite B, see Suite B Cryptography (https://en.wikipedia.org/wiki/NSA_Suite_B_Cryptography).

OpenText eDirectory allows you to separately configure the following modules in Suite B modes:

Module	Description
NPKI (OpenText certificate server)	Certificate Server provides public key cryptography services that are natively integrated into OpenText eDirectory and that allow you to mint, issue, and manage both user and server certificates. These services allow you to protect confidential data transmissions over public communications channels such as the Internet. When you configure the Certificate Server in Suite B mode, Certificate Server adheres to RFC 5759 that specifies the base profile for Suite B certificates and Certificate Revocation List (CRL). For more information, see "Enabling Suite B on the Certificate Server" on page 473.
LDAP and HTTP Services	The LDAP service is a server application that lets LDAP clients access information stored in OpenText eDirectory. It provides cross-platform monitoring and diagnostic capability to all servers in your eDirectory tree using the HTTP service. When you configure these services in Suite B mode, they include support for ECDSA certificates and enforce use of TLS 1.2 and Suite B ciphers as specified in RFC 6460. For more information, see "Configuring LDAP and HTTP Services to Use ECDSA Certificates and Suite B Ciphers" on page 474.

Module	Description
NICI	NICI is the cryptography module that provides keys, algorithms, various key storage and usage mechanisms, and a large-scale key management system. To help applications securely store and transfer data and keys, NICI provides three types of keys - Key Storage key, NICI Security Domain Infrastructure (SDI) key, and Session key. When you configure a server in Suite B mode, NICI secures sensitive data in the tree by using the 256-bit AES keys. For example, passwords, Challenge-Response data. Upgrading to NICI 3.0 automatically re-creates the key storage key and session key to adhere to Suite B. OpenText eDirectory uses NICI SDI key, also called tree key, to securely wrap keys that in turn encrypt data for local or remote storage allowing servers in the tree to unwrap the key. The data remains secure in conjunction with OpenText eDirectory rights. The tree key is available to all servers in the tree. To access the same data, multiple servers use the same NICI SDI key. Therefore, this key is not automatically created with NICI 3.0 installation. You need to manually create this key. For more information, see “Creating an AES 256-Bit SDI Key” on page 476.
Background authentication mechanism	Provides standards-based background authentication mechanism based on TLS 1.2 for single sign-on authentication with OpenText eDirectory. For more information, see “Enabling Background Authentication” on page 477.

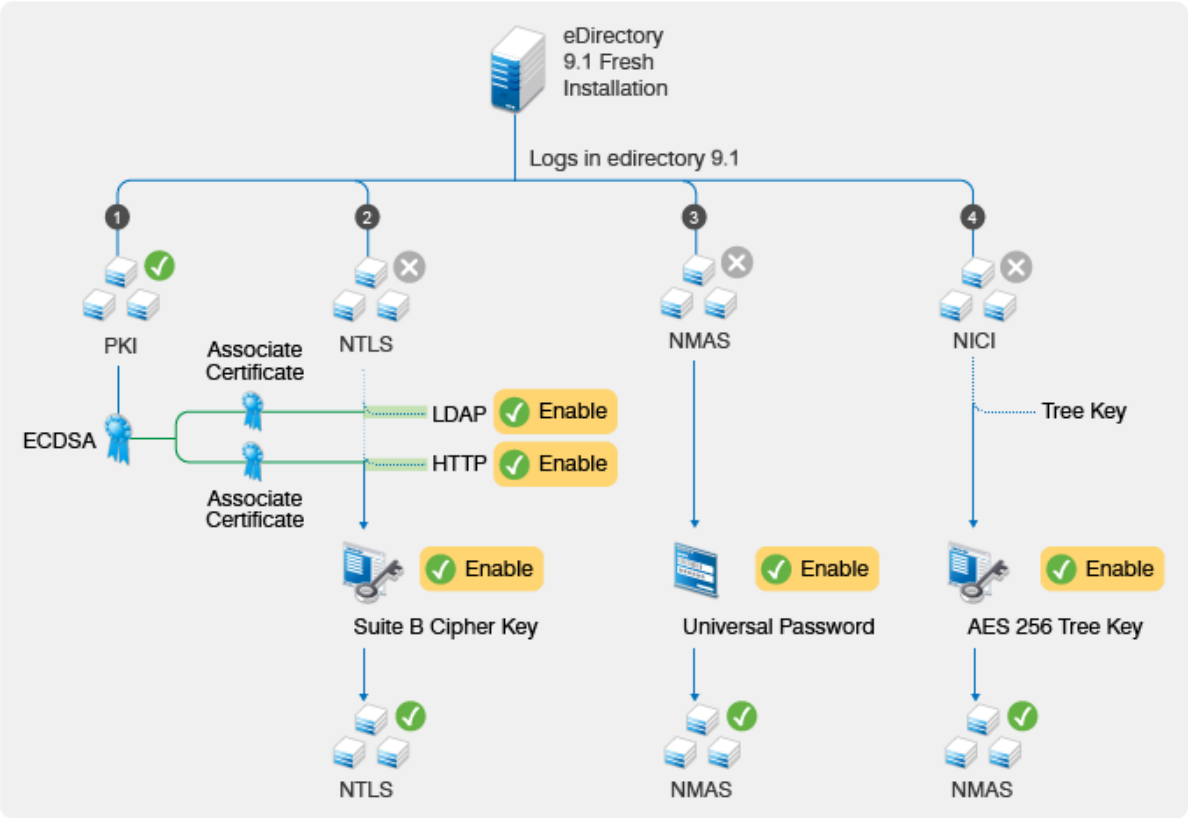
The following sections describe information about configuring OpenText eDirectory modules in Suite B modes:

- “Enabling Suite B in a New Installation” on page 472
- “Configuring Suite B on Existing Servers” on page 477

Enabling Suite B in a New Installation

Figure 16-1 shows the sequence of tasks for enabling Suite B on OpenText eDirectory components in a new installation.

Figure 16-1 *Enabling Suite B in a New Installation*



Enabling Suite B on the Certificate Server

When you configure a new tree, the Certificate Server creates a self-signed ECDSA certificate on P-384 curve for the tree Certificate Authority (CA) in addition to the traditional RSA certificates. If you add new servers to the tree or upgrade old servers to OpenText eDirectory 9.2, the Certificate Server issues the ECDSA certificates to these servers.

NOTE: By default, OpenText Certificate Server creates the ECDSA certificates on P-384 curve. However, you can also create server certificates on P-256 curve.

It is possible to use only the ECDSA certificates without enabling Suite B. Enabling Suite B is an additional step for adhering to RFC 5759.

To configure the CA Certificate Server to operate in the Suite B mode, perform the following steps:

1. Configure Enhanced Background Authentication for the CA Certificate Server.

   For more information, see "Enabling Background Authentication" on page 477.
2. Launch OpenText Identity Console, click **Certificate Management** tile > **CA Management**.
3. Click **CA Configuration** .
4. On the CERTIFICATE SELF PROVISIONING page > select check box **Enable Suite B Mode**.
5. Click **Apply** > **OK**.

When the CA Certificate Server is in Suite B mode, the CA does not allow you to create RSA certificates. Also, server self-provisioning does not generate RSA certificates any longer. If you plan to add new servers, ensure that the servers are configured to run Enhanced Background Authentication.

When all servers in the tree or any external services connecting to the tree start using the ECDSA server certificates, you can revoke and delete the RSA certificates because they are not needed any more.

NOTE: The Follow CA's Algorithm feature that was introduced in OpenText eDirectory 8.8.8 Patch 6 is no longer available with OpenText eDirectory 9.0 or later. Instead, OpenText eDirectory 9.2 servers use SHA-256 algorithm for RSA certificates and SHA-384 for ECDSA certificates by default.

Configuring LDAP and HTTP Services to Use ECDSA Certificates and Suite B Ciphers

OpenText Transport Layer Security (NTLS) supports TLS 1.2 and Suite B cryptographic algorithms through the FIPS compliant OpenSSL module. The FIPS compliant OpenSSL module is used by OpenText eDirectory components such as LDAP, httpstk (iMonitor), and NCP engine. For more information, see Operating OpenText eDirectory in FIPS Mode in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

Before enabling a Suite B mode on a server, ensure that the server has ECDSA certificates and LDAP clients, LDAP browsers, and web browsers in the OpenText eDirectory environment support TLS 1.2, ECDSA certificates, and Suite B ciphers. To configure LDAP and HTTPS interfaces in a Suite B mode, enable the interfaces with the desired Suite B mode and associate an appropriate ECDSA server certificate to them. Repeat this procedure for each eDirectory server in the tree. To view the cipher level and ECDSA server certificate for a server, use the server's LDAP and httpstk configuration objects: ldapServer and httpServer.

To configure the LDAP server in Suite B mode:

1. Launch OpenText identity console.
2. Log into OpenText eDirectory as an administrator with the appropriate rights.
3. Click **LDAP Servers** object you want to configure in Suite B mode.
4. Click **Information**.
5. In the **Server Certificate** drop down, select SSL `Elliptic Curve CertificateDNS` that you want to use with the LDAP server object.
6. Click **Configuration** tile.
7. Depending on the Suite B mode that you want to enable for the LDAP server object, select a value from the **Bind Restrictions for Cipher** drop-down list.

Bind Restrictions for Cipher	Cipher Suite	Description
Use SuiteB Cipher (128-bit)	◆ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ◆ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Enables Suite B mode operation by using 128-bit level of security. When you select this option, OpenText eDirectory permits both 128-bit and 192-bit level of security by peers (any LDAP clients). You can use either ECDSA 256 or ECDSA 384 certificate with this option.
Use SuiteB Cipher (128-bit only)	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Enables Suite B mode operation by using 128-bit level of security. When you select this option, OpenText eDirectory does not allow 192-bit level of security by peers (any LDAP clients). All certificates in a certificate chain should use ECDSA keys on P-256 curve. This is mandatory for servers and applicable for clients if client certificate validation is enabled.
Use SuiteB Cipher (192-bit)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Enables Suite B mode operation by using 192-bit level of security. When you select this option, OpenText eDirectory permits only 192-bit level of security by peers (any LDAP clients). All certificates in a certificate chain should use ECDSA keys on P-384 curve. This is mandatory for servers and applicable for clients if client certificate validation is enabled.

OpenText eDirectory allows you to use combination values of ldapbindrestrictions and cipher levels. For more information, see Table 14-1 on page 378.

8 Click **Save,** then click **OK**.

9 For the changes to take effect, do one of the following:

- Restart OpenText eDirectory.
- Unload and load the LDAP server.

To configure the HTTPS interface in Suite B mode:

1 Log in to OpenText Identity Console with the OpenText eDirectory tree as an administrator with the appropriate rights.

2 go to **Object Management**.

3 From **Type** drop-down, select the `httpServer` object you want to modify, and click **Search**.

4 Click `Http Server`, and select `httpBindRestrictions` from the **Valued Attributes** list.

5 Depending on the Suite B mode that you want to enable for the `httpserver` object, change the value to 4,5, or 6 in the dialog that displays.

Bind Restrictions for Cipher	Cipher Suite	Description
4 - Use SuiteB Cipher (128-bit)	◆ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ◆ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Enables Suite B mode operation by using 128-bit level of security (Suite B Cipher 128-bit). When you select this option, OpenText eDirectory permits both 128-bit and 192-bit level of security by clients (web browsers). You can use either ECDSA 256 or ECDSA 384 certificate with this option.
5 - Use SuiteB Cipher (128-bit only)	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Enables Suite B mode operation by using 128-bit level of security (Suite B Cipher 128-bit only). When you select this option, OpenText eDirectory does not allow 192-bit level of security by clients (web browsers). All certificates in a certificate chain should use ECDSA keys on P-256 curve. This is mandatory for servers and applicable for clients if client certificate validation is enabled.
6 - Use SuiteB Cipher (192-bit)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Enables Suite B mode operation by using 192-bit level of security (Suite B Cipher 192-bit). When you select this option, eOpenText Directory permits only 192-bit level of security by clients (web browsers). All certificates in a certificate chain should use ECDSA keys on P-384 curve. This is mandatory for servers and applicable for clients if client certificate validation is enabled.

6 Select `httpKeyMaterialObject` from the **Valued Attributes** list and click Edit.

7 Browse to and select the Elliptic Curve certificate that you want to use with the HTTPS interface and click **OK**.

8 Click **Save.**

9 Click **OK**.

10 Restart eDirectory for the changes to take effect.

Creating an AES 256-Bit SDI Key

By default, the NICI SDI key is a 3DES key. However, to support Suite B modes, you need to manually create the AES 256-bit NICI SDI key. Create this key only when all servers in the tree are OpenText eDirectory 9.0 or above.

When a server holding the writeable replica of KAP.Security container is upgraded to OpenText eDirectory 9.2, the PKI health check will create a W1 object in this container. When all servers in the tree are upgraded to eDirectory 9.2, the tree administrator can create the AES 256-bit NICI SDI key.

To create the AES 256-bit NICI SDI key, follow the instructions from Creating an AES 256-Bit Tree Key in the NICI Administration Guide.

Re-encrypting Data with AES 256-Bit NICI SDI Key

NMAS uses the NICI SDI key to securely store passwords and Challenge-Response configuration (questions and answers). NMAS also has a secret store for the user and method specific configuration that uses NICI SDI key. To re-encrypt passwords for multiple users in large deployments, use the `diagpwd` utility. For more information, see "Universal Password Diagnostic Utility" on page 741.

IMPORTANT: If your OpenText eDirectory environment includes servers with the versions prior to OpenText eDirectory 9.0, those servers will not be able to decrypt passwords or secret data which are encrypted with AES 256-bit tree key resulting in a failed login to these servers.

Enabling Background Authentication

OpenText eDirectory provides a strong authentication mechanism that verifies the identity of users who request to access it. For more information about enhanced background authentication, see Chapter 17, "Enabling Enhanced Background Authentication," on page 479.

Configuring Suite B on Existing Servers

To enable Suite B on the existing servers in your OpenText eDirectory tree, perform the following actions:

1. Upgrade the server acting as CA to OpenText eDirectory 9.3.

   When the CA server is upgraded, the server creates the ECDSA self-signed CA certificate. When other servers are upgraded to OpenText eDirectory 9.3, the new CA issues ECDSA certificates to these servers.

2. Upgrade the desired servers in the tree to OpenText eDirectory 9.3.

   The upgrade process generates ECDSA certificates for the upgraded servers. You must use these certificates for enabling the LDAP and HTTP protocol stack interfaces to Suite B mode. For more information, see "Configuring LDAP and HTTP Services to Use ECDSA Certificates and Suite B Ciphers" on page 474.

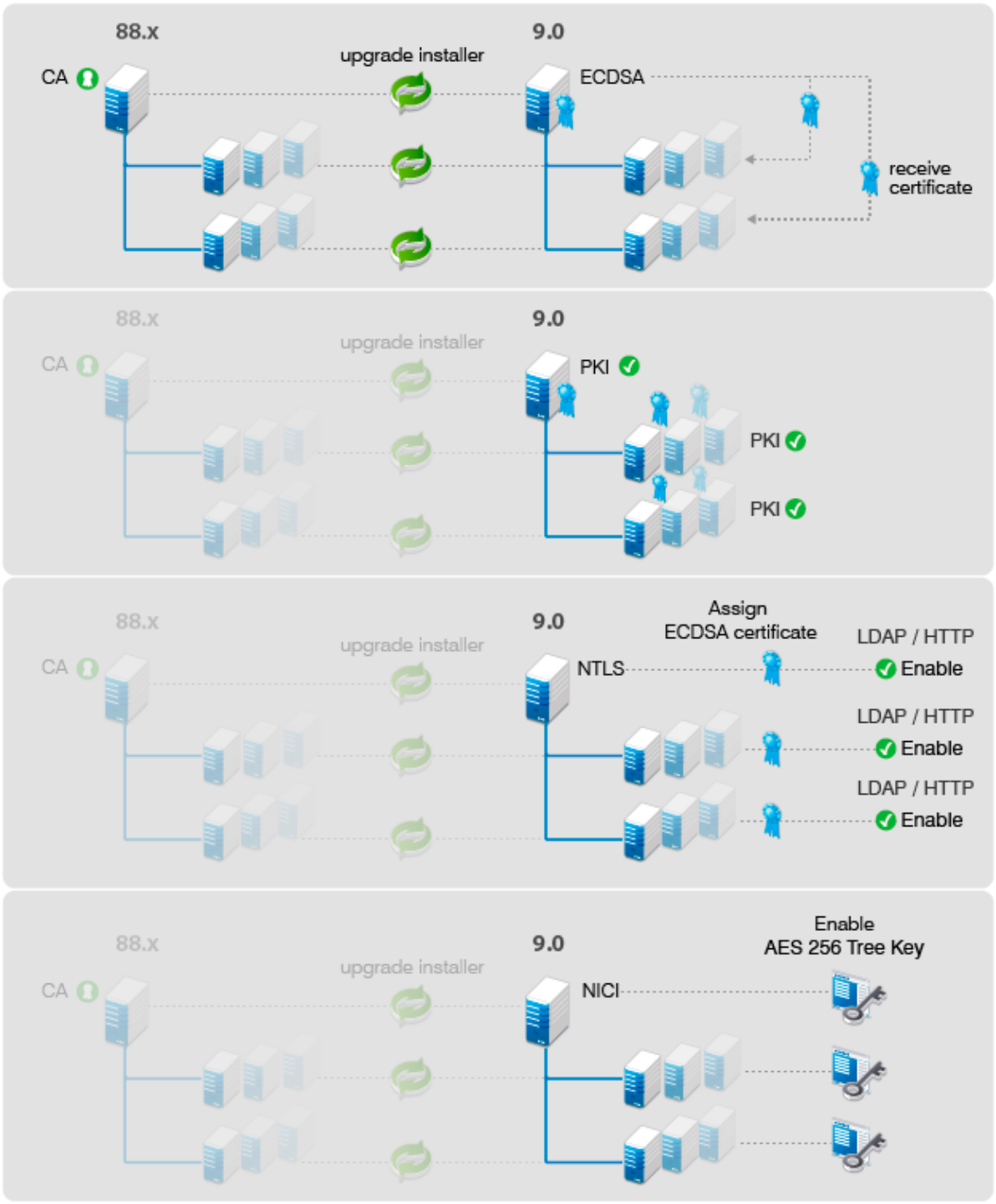
3. Create an AES 256-Bit SDI Key. For more information, see "Creating an AES 256-Bit SDI Key" on page 476.

4. Re-encrypt the data with the AES 256-bit NICI SDI key. For more information, see "Creating an AES 256-Bit SDI Key" on page 476.

5. Configure background authentication. For more information, see "Enabling Enhanced Background Authentication" on page 479.

Figure 16-2 shows the sequence of tasks for enabling Suite B when you upgrade OpenText eDirectory.

Figure 16-2 *Enabling Suite B When OpenText eDirectory is Upgraded*



17 Enabling Enhanced Background Authentication

OpenText eDirectory provides a strong authentication mechanism that verifies the identity of users who request to access it. Authentication includes two phases:

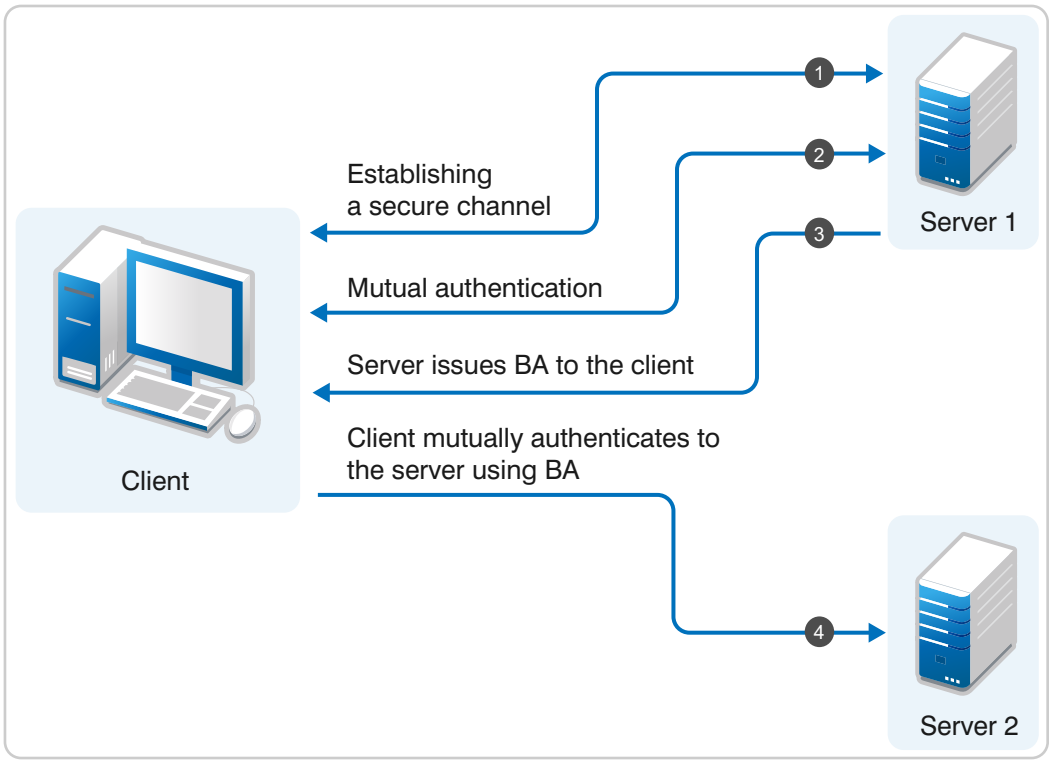
- Login
- Background authentication (BA)

When a user logs in, NetIQ Modular Authentication Service (NMAS) verifies the user's long-term credentials, such as password, and issues BA material to the user.

While authenticating to another server in the tree, the user uses this BA material. This single sign–on feature of OpenText eDirectory allows the user to authenticate to any server in the tree without providing long-term credentials again.

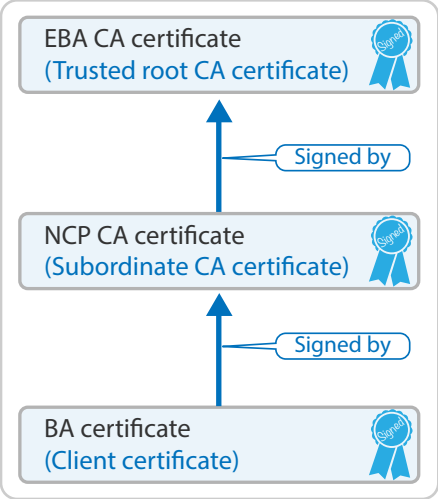
OpenText eDirectory 9.0 introduces a standards-based BA protocol that enables you to overcome the limitations of the proprietary BA protocol. This protocol is known as Enhanced Background Authentication (EBA). When EBA is used, NMAS issues the users an X.509 certificate as the BA material and the BA protocol uses TLS version 1.2 for mutual authentication.

Figure 17-1 *EBA Process*



In an EBA-enabled OpenText eDirectory tree, the EBA CA is the trusted root certifying authority for EBA. EBA CA uses a self–signed certificate. You can configure one of the servers in the tree with a writable replica of the tree root partition as the EBA CA. Usually, the first server configured in the tree hosting the writable replica of the tree-root partition and configured with EBA acts as an EBA CA. Alternatively, you can configure any OpenText eDirectory 9.2 server in the tree with a writable replica of the tree–root partition to act as EBA CA.

Figure 17-2 *EBA Certificate Authority*



Each EBA-enabled server in the tree becomes a CA subordinate to EBA CA and is called NCP CA. After login, NMAS returns a BA certificate issued by NCP CA to the logged-in user.

NOTE: Any object that logs into OpenText eDirectory must have the OID set in the schema for all the naming attributes in the object DN.

To use EBA to authenticate to an OpenText eDirectory server, a client needs the EBA CA certificate of the tree. To obtain the EBA CA certificate, use the ebaclientinit utility. This is a new command line utility packaged with OpenText eDirectory 9.0 and above. This utility downloads the EBA CA certificate of the tree and saves it in a file named `.eba.p12`. This file is present in the user's home directory on Linux (`$HOME`) and in the user's profile directory (`%USERPROFILE%`) on Windows.

NOTE: For EBA to function properly, synchronize the time on all EBA-enabled servers and clients in your OpenText eDirectory environment.

When you run the ebaclientinit utility for multiple trees, the utility adds the EBA CA certificates associated with the OpenText eDirectory trees to the `.eba.p12` file. To obtain the EBA CA certificate for each OpenText eDirectory tree, run the ebaclientinit utility once for that tree.

Enabling EBA

This section helps you enable EBA on OpenText eDirectory. Depending on your type of installation, follow the instructions from one of the following sections:

- “Enabling EBA on an OpenText eDirectory Tree” on page 481
- “Enabling EBA on an OpenText eDirectory Server” on page 482
- “Disabling EBA on an OpenText eDirectory Server” on page 483

Enabling EBA on an OpenText eDirectory Tree

- “Enabling EBA on a New OpenText eDirectory Tree” on page 481
- “Enabling EBA on an Existing Tree” on page 481

Enabling EBA on a New OpenText eDirectory Tree

To enable EBA on a new tree, perform one of the following actions depending on your platform:

- **Linux:** To enable EBA while configuring a new OpenText eDirectory tree, run the `ndsconfig` command with `--configure-eba-now` argument at the command line.

  For example: `ndsconfig new --configure-eba-now yes`

  If this argument is not passed to the command, you are prompted to enable EBA. Based on your preference, enter **yes** or **no** at the prompt.
- **Windows:** The installation program provides the option of enabling EBA during OpenText eDirectory configuration. To enable EBA, select the **Enable EBA** option during configuration.

NOTE: OpenText eDirectory does not allow you to configure `ebaext.dlm` and `ebassl_srv.dlm` modules on an EBA-enabled server for auto-startup because the DS module automatically loads them when you enable EBA on an OpenText eDirectory server.

If you attempt to load `ebaext.dlm` and `ebassl_srv.dlm` modules on a non EBA-enabled server, the modules might successfully load, but the EBA functionality will not work.

Enabling EBA on an Existing Tree

To enable EBA on an existing OpenText eDirectory tree, perform one of the following actions depending on your platform:

- **Linux:** Run the `ndsconfig upgrade` command with `--configure-eba-now` argument on one of the servers having a writable replica of the tree-root partition.

  For example: `ndsconfig upgrade --configure-eba-now yes`
- **Windows:** Run `eDirectory_910_Windows_x86_64.exe` from the OpenText eDirectory 9.2 installation folder on one of the servers having a writable replica of the tree-root partition and select the **Enable EBA** option during the OpenText eDirectory configuration.

Enabling EBA on an OpenText eDirectory Server

When you enable EBA on an OpenText eDirectory server, a Certificate Signing Request (CSR) is sent to the EBA CA. The EBA CA validates the CSR, performs access control checks, and then issues the NCP CA certificate to the server. Before enabling EBA on a server, ensure that:

- (Mandatory) A writable replica of the partition containing the administrator DN is present on a EBA-enabled server in the tree.
- (Optional) A writable replica of the partition containing the server object is present on a EBA-enabled server in the tree. If the server does not meet this condition, the EBA CA saves the CSR and does not issue the NCP CA certificate. This causes OpenText eDirectory configuration to fail and requires the administrator to approve the CSR using the EBA plug-in of OpenText Identity Console. For more information, see "Managing the EBA CA by Using OpenText Identity Console" on page 485. After the CSR is approved, configure EBA by running the `ndsconfig upgrade` command. For example, `ndsconfig upgrade --configure-eba-now yes`
- "Enabling EBA When a New Server is Added" on page 482
- "Enabling EBA on a Configured Server" on page 482

Enabling EBA When a New Server is Added

To enable EBA when a new server is added to a tree, perform one of the following actions depending on your platform:

- **Linux:** Run the `ndsconfig add` command with `--configure-eba-now yes` argument.

  For example: `ndsconfig add --configure-eba-now yes`
- **Windows:** Run `eDirectory_910_Windows_x86_64.exe` from the OpenText eDirectory 9.3 installation folder and select the **Enable EBA** option during the eDirectory configuration.

Enabling EBA on a Configured Server

To enable EBA on a configured server, perform one of the following actions depending on your platform:

- **Linux:** Run the `ndsconfig upgrade` command with `--configure-eba-now yes` argument.

  For example: `ndsconfig upgrade --configure-eba-now yes`
- **Windows:** Run `eDirectory_910_Windows_x86_64.exe` from the OpenText eDirectory 9.3 installation folder and select the **Enable EBA** option during the OpenText eDirectory configuration.

IMPORTANT: In addition to the server acting as the EBA CA, OpenText recommends that you must have at least one more EBA-enabled server containing the Read/Write replica of the tree-root partition. If the server acting as EBA CA goes down, the other EBA-enabled server can be configured to act as EBA CA. For more information, see "Moving the EBA CA Role to a New Server" on page 486.

Disabling EBA on an OpenText eDirectory Server

To disable EBA on a configured server, perform one of the following actions depending on your platform:

- **Linux:**
  - Run the following commands to restart the OpenText eDirectory server with EBA disabled:

    ```
    ndsmanage stopall
    export DISABLE_EBA=true
    ndsmanage startall
    ```

  - Run the following commands to restart the OpenText eDirectory server with EBA enabled:

    ```
    ndsmanage stopall
    unset DISABLE_EBA
    ndsmanage startall
    ```

  NOTE: You must add all the environment variables required for the OpenText eDirectory service in the env file located in the `/etc/opt/novell/eDirectory/conf` directory on RHEL 7.x and SLES 12.x platforms.

- **Windows:** Go to **Control Panel** > **System** > **Advanced System Settings** > **Environment Variables** > **System Variables** > **New**. Add a new variable called `DISABLE_EBA` with value 1 and restart the server.

IMPORTANT: You must disable EBA only for troubleshooting purpose. If EBA is disabled on an OpenText eDirectory server which is acting as EBA CA for 7 days or more, the EBA functionality on the OpenText eDirectory tree will be broken. For more information, see TID 7017232.

Viewing Information About EBA

The following lists different tools and utilities that provide information about different aspects of an EBA-enabled server.

Utility	Description
ndstrace	Provides information about EBA operations such as EBA request handling, issue of NCP CA certificates. To view this information: On Linux, enable the EBA tag of ndstrace. On Windows, load the dstrace module enabled with EBA tag in `NDSCons.exe`.
ndsd.log, dhost.log	Provide information about EBA start and stop messages.To view this information: On Linux, see the `ndsd.log` file of the server where you configured EBA. On Windows, this information is logged in the `dhost.log` file.

Utility	Description
ndscheck	Provides the status of EBA on a server - whether the server is EBA–enabled. If the server is EBA-enabled, the output of the command displays information such as NCP CA certificate validity, whether the server is acting as EBA CA, and so on. When you run the `ndscheck` command remotely, ensure that the EBA CA certificate of the tree has been downloaded to the local computer.
ndslogin	Provides troubleshooting information about EBA configuration. For troubleshooting EBA configuration, log in to OpenText eDirectory by using the `ndslogin` command with -c argument. For example: `ndslogin <admin DN> –p <password> –c` For a successful login, ensure that the EBA CA certificate of the tree has been downloaded to the local computer.
schema.log	The `schema.log` file of the server where EBA is configured contains information about schema extension for EBA.
nioutput.log	Specifies whether EBA was selected during OpenText eDirectory configuration.
iMonitor	OpenText iMonitor lets you monitor your EBA-enabled servers for the following information: ◆ Determine whether a server is EBA–enabled by looking at the EBA enabled parameter under the Connection Information tab of the Agent Configuration page. If the value of this parameter is true, the server is EBA–enabled. Otherwise, the value for this parameter is false. ◆ Display the same debugging information that you can view in ndstrace on Linux.The trace configuration page includes a tag for EBA for displaying this information. ◆ View information about the validity of the EBA CA certificate, NCP CA certificate, and the EBA–enabled status of the server on the Agent Health page. These items appear Green if the certificate is valid and the EBA attributes are unaffected. ◆ View the EBA Request verb on an EBA–enabled server in the Verb Statistics page. ◆ View the iMonitor Health Check Agent page for the following: ◆ Whether the server is EBA-enabled ◆ Whether the server hosts EBA CA ◆ Whether the EBA CA certificate is valid ◆ Whether the NCP CA certificate is valid

Managing the EBA CA by Using OpenText Identity Console

To access OpenText eDirectory from the **EBA** tile of OpenText Identity Console, the EBA CA certificate must reside in the EBA trusted certificate store of OpenText Identity Console.

To open the EBA CA Management page, log in to OpenText Identity Console, on the home page click **EBA** tile. The **EBA CA Management** page opens.

The **EBA CA management** page includes the following tabs to manage different aspects of EBA CA:

- **General:** Displays the IP address of EBA CA and its certificate.
- **Certificates Issued:** Displays the NCP CA certificates along with their IP address and port.

  To revoke a certificate, select the certificate and click **Revoke**. Use this option only in extreme situations, because the server owning the NCP CA certificate will become non-functional when you revoke its certificate. Usually, revoking the certificate becomes necessary when a server is compromised.
- **CSR:** Lists the pending certificate signing requests for administrator approval. To approve a certificate signing request, select the certificate from the list and click **Approve**.

Restrictions in OpenText eDirectory Operations When EBA Is Enabled

If a server holding the Master Read/Write replica of a partition is EBA-enabled, the partition is considered EBA–enabled. OpenText eDirectory does not allow any operation that makes a partition non–EBA enabled. On an EBA-enabled server, OpenText eDirectory imposes the following restrictions on the partition and replica operations:

- "Restrictions on Changing Replica Types" on page 485
- "Restrictions on Changing the Master of a Partition" on page 486
- "Restrictions on Merging Partitions" on page 486
- "Restrictions on Reconfiguring a Server Enabled with EBA" on page 486

Restrictions on Changing Replica Types

- If the server hosting EBA CA holds the master replica of the tree root partition, OpenText eDirectory does not allow you to change its replica type.
- If EBA CA is hosted on a server that holds the Read/Write replica of the tree root partition, ensure that you do not change the replica type to anything other than the master. Changing the replica type to Read-Only/Filtered Read-Only might break the EBA functionality in the entire tree. OpenText eDirectory enforces this restriction if the OpenText eDirectory server holding the master replica of the tree root partition is EBA-enabled.

NOTE: In a mixed environment where you have older and OpenText eDirectory 9.2 servers, you might be successful in changing the replica types; however, doing this can break the EBA functionality.

Restrictions on Changing the Master of a Partition

If the master replica of a partition is present on an EBA-enabled server, the following operations will fail:

- Transferring the role of master to a non EBA-enabled server.
- Transferring the role of master to any other server, if the server is acting as EBA CA.

Restrictions on Merging Partitions

Do not merge two partitions when the parent partition is not EBA–enabled and the child partition is EBA–enabled. Doing this might break the EBA functionality.

Restrictions on Reconfiguring a Server Enabled with EBA

When OpenText eDirectory is configured on a server in EBA-enabled mode, this setting is saved in the `n4u.server.eba_enabled` parameter in the `nds.conf` file. When OpenText eDirectory is deconfigured on this server and the server configured again, EBA is turned on by default. To configure the server in a non-EBA mode, remove this parameter from the nds.conf file before configuring OpenText eDirectory on the server.

Backing Up an EBA Enabled Server

For backing up an EBA-enabled OpenText eDirectory server, follow the instructions from Backing Up and Restoring OpenText eDirectory. Ensure that you select NICI and stream attributes while taking an incremental or a full backup of an EBA-enabled server. Otherwise, you cannot restore the server.

Moving the EBA CA Role to a New Server

If the server acting as EBA CA is down, OpenText eDirectory provides the flexibility of moving the EBA CA role to a different EBA-enabled server in the tree. Before moving the EBA CA role to new server, ensure that the new server:

- Is EBA-enabled.
- Has a writable replica of the tree-root partition where the replica was already created when the EBA CA went down.

To transfer the role of EBA CA to the new server on Linux operating systems, run the following command from your bash shell on the new server:

```
ndstrace -c "config ebassl_srv seize_ebaca"
```

OpenText eDirectory displays a success message indicating that the EBA CA role is transferred to the new server. If you try this operation while the original server is still functional, the operation fails.

On Windows, perform the following steps:

1. Open `ndscons.exe`.
2. Click **Start > Settings > Control Panel > OpenText eDirectory Services**.

3 On the **Services** tab, scroll to `ebassl_srv.dlm`, then enter `seize_ebaca` in the **Startup Parameters** field.

4 Click **Configure**.

To view messages about the EBA CA role transfer, run `dstrace.dlm` with EBA tag enabled when the EBA CA role transfer operation is running. DSTrace displays the appropriate message depending on the success or failure of the operation. If you try this operation while the original server is still functional, the operation fails.

NOTE: ◆To determine whether the EBA CA role was successfully seized or not, run ndscheck on the new server. If the ndscheck output shows EBACA=`true`, the new server is now the EBA CA of the tree.

- If the server hosting EBA CA is down, designate some other server in the replica ring of the tree-root partition as EBA CA. If the server that went down had a master replica of the tree-root partition, it is recommended to transfer the master role to the new server acting as EBA CA. To transfer the master role, follow the instructions from "Repairing Replicas" on page 318.

---

18 SNMP Support for OpenText eDirectory

The Simple Network Management Protocol (SNMP) is the standard operations and maintenance protocol for the Internet for exchanging management information between the management console applications and managed devices. Management console application are application such as IBM Tivoli NetView or Solstice SunNet Manager. The managed devices includes hosts, routers, bridges, and hubs and also network applications like OpenText eDirectory.

This chapter describes SNMP services for OpenText eDirectory. It contains the following topics:

- "Definitions and Terminology for SNMP" on page 489
- "Understanding SNMP Services" on page 490
- "OpenText eDirectory and SNMP" on page 492
- "Installing and Configuring SNMP Services for OpenText eDirectory" on page 494
- "Monitoring OpenText eDirectory Using SNMP" on page 500
- "Troubleshooting" on page 526

Definitions and Terminology for SNMP

The following tables contain terminologies used in this chapter.

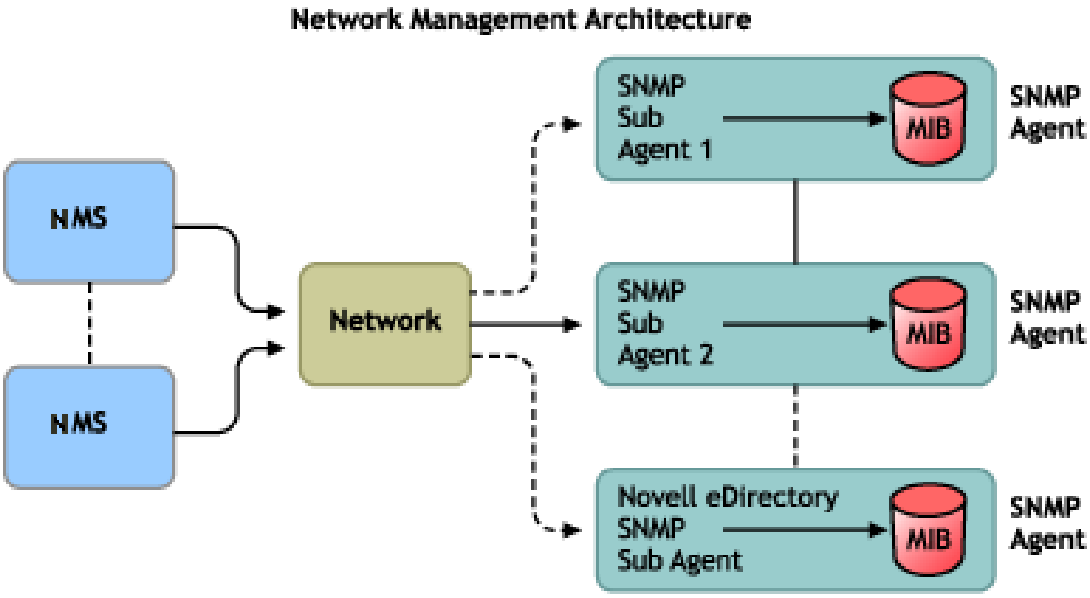
Terminology	Definition
EMANATE	Enhanced Management Agent Through Extensions is a product from SNMP Research International, Inc.
SNMP	Simple Network Management Protocol is used to exchange data about the network activity.
NAA	Native Agent Adapter
NMS	Network Management Station
MA	Management Agent
SA	Subagent
MIB	Management Information Base
NCP	NetWare/Novell Core Protocol
NMA	Network Management Application
edir.mib	OpenText eDirectory server Monitoring MIB, which has MIB objects and traps relevant to OpenText eDirectory.
traps	Alerts generated by agents on a managed device when OpenText eDirectory events occur on the server. These conditions are defined in the Management Information Base (MIB) provided by OpenText.

Understanding SNMP Services

SNMP is based on a manager/agent architecture. The architecture of network management with SNMP includes the following elements:

- Network Management Station (NMS)
- Managed Device
- Master Agent
- Subagent
- Management Information Base (MIB)
- Network Management Protocol

Figure 18-1 *Network Management Architecture*



Network Management Station

The network management station is a workstation with one or more network management applications installed, to graphically show information about managed devices.

NMS features:

- Provides the user interface to the entire network management system, thus providing a powerful, flexible and easy to use tool for network management
- Allows you to perform SNMP Get, Get Next, SNMP Get Response and Set operations. NMS also allows you to capture SNMP Traps sent from managed devices on the network.
- Monitors one or more network management applications (NMA) simultaneously. NMS has facilities to graphically show information about managed devices, table viewing, and logging.
- Allows you to compile the MIB file using the MIB compiler present in the NMS.

Managed Devices

A managed device is any device that has SNMP installed on it. A managed device could be a host, router, bridge, hub, etc. NMS can monitor and communicate with managed devices.

The information between the NMS and the managed device is transferred through two types of agents: subagent and master agent.

Subagent

The subagent gathers information about the managed device and passes the information to the master agent.

Master Agent

The master agent exchanges information between the various subagents and the NMS. The master agent runs on the same host machine as the subagents with which it communicates.

Management Information Base

SNMP exchanges network information in the form of protocol data units (PDUs). PDUs contain information about variables stored on the managed device. These variables are known as managed objects and have values and titles that are reported to the NMS. All managed objects are defined in the Management Information Base (MIB). MIB is a virtual database with a tree-like hierarchy.

SNMP Network Management Protocol

The basic functions of SNMP are listed in the following table.

Function	Description
Get	Used by the manager to request information from an agent.
Get Next	Used by the manager to obtain information from an array or a table.
Get Response	Used by the queried agent to satisfy a request made by the manager.
Set	Used by the manager to modify the value of the variable which resides on the agent's MIB.
Trap	Used by the agent to notify the manager that a certain event has occurred.

For more information about SNMP, refer to the following Web sites:

- NET-SNMP Home Page (http://net-snmp.sourceforge.net)
- SNMP FAQ (http://www.faqs.org/faqs/snmp-faq/part1)
- RFC 1157 (http://www.ietf.org/rfc/rfc1157.txt)
- SNMPInfo (https://metacpan.org/pod/SNMP::Info)
- SNMP RFC Standard MIBs and Informative Links (https://www.snmp.com/protocol/snmp_rfcs.shtml)
- RFC 2605 (http://www.ietf.org/rfc/rfc2605.txt?number=2605)

OpenText eDirectory and SNMP

OpenText eDirectory can store and manage millions of objects, such as users, applications, network devices, and data. With the increase in objects, the need to track down the additions and modifications to the OpenText eDirectory increases. SNMP renders a solution to this problem by helping you monitor OpenText eDirectory servers and thus keep track of the changes.

Benefits of SNMP Instrumentation on OpenText eDirectory

- Real time monitoring for an OpenText eDirectory server
- Monitoring of OpenText eDirectory from any third party SNMP MIB browser
- Tracking the status of OpenText eDirectory to verify normal operations
- Spotting and reacting to potential problems once they are detected
- Configuring traps and statistics for selective monitoring
- Plotting a trend on the access of OpenText eDirectory
- Storing and analyzing historical data that has been obtained through SNMP
- SNMP Get, GetNext request support for statistics
- Using SNMP native master agent on all the platform

Understanding How SNMP Works with OpenText eDirectory

SNMP implementation on OpenText eDirectory provides useful OpenText eDirectory information on statistics on the accesses, operations, errors, and cache performance. Traps on the occurrence of events can also be sent with SNMP implementation. Traps and statistics are defined in the MIB.

NOTE: You might have to access the encrypted attributes only over a secure channel, if you have specified that you always need a secure channel to access these attributes. For more information, refer to “Encrypted Attributes” on page 293.

Directory Service Monitoring MIB

The OpenText eDirectory MIB defines statistics and traps to monitor OpenText eDirectory. This MIB is assigned the following id:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).novell(23).mibDoc(2).ndsMIB(98)
```

Statistics

The OpenText eDirectory MIB is divided into four distinct tables of managed objects:

- **The Cache Database Statistics Table - ndsDbCacheTable:** Contains a description of the directory servers as well as summary statistics on the entries cached by these servers.
- **The Config Database Statistics Table - ndsDbConfigTable:** Contains a description of the directory servers as well as summary statistics on the entries configured by these servers.

- **The Protocol Statistics Table - ndsProtoIfOpsTable:** Provides summary statistics on the accesses, operations, and errors for each application protocol interface of a directory server.
- **The Interaction Statistics Table - ndsServerIntTable:** Keeps track of the last “N” directory server with which the monitored directory has interacted or attempted to interact. “N” is a locally defined constant.

NOTE: For more information on statistics, see “Statistics” on page 522.

Traps - ndsTrapVariables

The OpenText eDirectory MIB defines 119 traps. Out of this, 117 traps map to OpenText eDirectory events and 2 additional traps `ndsServerStart` and `ndsServerStop` are directly generated by the SNMP subagent. These two traps cannot be configured.

NOTE: For more information on traps, see “Traps” on page 501.

For more information on statistics and traps, see `edir.mib`.

`edir.mib` is located in the following directories:

Windows: *`install_directory`*`\SNMP`
Linux: `/etc/opt/novell/eDirectory/conf/ndssnmp/`

SNMP Group Object

The SNMP group object is used to set up and manage the OpenText eDirectory SNMP traps. During installation, an SNMP group object named “SNMP Group - *server_name*” is created (where *server_name* is the name of the server on which SNMP services for OpenText eDirectory are installed). The SNMP group object is created in the same container as the server object. This SNMP configuration utility is used to configure SNMP traps.

On Windows

To create an SNMP group object, enter the following command:

`rundll32 snmpinst, snmpinst -c <createobj> -a <userFDN> -p <password> -h <hostname or IP address>`

Parameter	Description
`-c <createobj>`	Trap command that specifies the creation of an object.
`-a <userFDN>`	Fully distinguished name of a user having administrative rights
`-p <password>`	userFDN password for authentication
`-h <hostname or IP address>`	DNS host name or IP address

Example:

```
rundll32 snmpinst, snmpinst -c createobj -a admin.mycontext -p mypassword -
h 160.98.146.26
```

To delete an SNMP group object, enter the following command:

```
rundll32 snmpinst, snmpinst -c <deleteobj> -a <userFDN> -p <password> -h
<hostname or IP address>
```

See the table above for more information.

Example:

```
rundll32 snmpinst, snmpinst -c deleteobj -a admin.mycontext -p mypassword -
h 160.98.146.26
```

On Linux

To create an SNMP group object, enter the following command:

```
ndsconfig add -m <modulename> -a <userFDN>
```

Example:

```
ndsconfig add -m snmp -a admin.mycontext
```

Installing and Configuring SNMP Services for OpenText eDirectory

SNMP service for OpenText eDirectory is installed when OpenText eDirectory is installed. You can modify the default configuration of SNMP services for OpenText eDirectory using OpenText Identity Console. For more information, see “Dynamic Configuration” on page 497.

A new object called SNMP Group-Object is added to the directory tree when OpenText eDirectory is installed. This object is used to set up and manage the OpenText eDirectory SNMP traps. See “SNMP Group Object” on page 493 for more information.

Installing SNMP after OpenText eDirectory Installation on Windows

If the SNMP service is not installed with OpenText eDirectory, the OpenText eDirectory install copies only the required SNMP subagent files and does not update the registry.

If you want to use SNMP services on OpenText eDirectory at a later point in time, you can install the SNMP service and update the registry using the following command:

```
rundll32 snmpinst, snmpinst -c createreg
```

Loading and Unloading the SNMP Server Module

The SNMP server module can be manually loaded and unloaded. By default, the SNMP server module loads automatically on all platforms. However, you can manually load the server module on Windows and Linux.

To load the SNMP server module, enter the following commands:

Server	Command
Windows	In the DHost (NDSCONS) screen, select **ndssnmp.dlm** > click **Start**.
Linux	In the DHost remote management page, to load the SNMP trap server click on the SNMP Trap Server for OpenText eDirectory action icon to start. or At the prompt, enter the following: `/opt/novell/eDirectory/bin/ndssnmp -l`

To unload the SNMP server module, enter the following commands:

Server	Command
Windows	In the DHost (NDSCONS) screen, select **ndssnmp.dlm**, then click **Stop**.
Linux	In the DHost remote management page, to unload the SNMP trap server, click the **SNMP Trap Server for OpenText eDirectory 9.2** action icon to stop. or At the prompt, enter the following: `/opt/novell/eDirectory/bin/ndssnmp -u`

Subagent Configuration

- "Static Configuration" on page 495
- "Dynamic Configuration" on page 497

Static Configuration

Static configuration is used before bringing up the subagent. You can manually configure it by editing the `ndssnmp.cfg` file on Windows or Linux. The `ndssnmp.cfg` file is located in the following directories:

Windows: *`install_directory`*`\SNMP\`

Linux: `/etc/opt/novell/eDirectory/conf/ndssnmp/`

NOTE: If changes are made to the `ndssnmp.cfg` file, the subagent must be restarted.

You can provide configuration information to the subagent such as the following:

- `INTERACTIVE` *`status`*

Where *status* is either on or off. If the status is on, you are prompted to enter the user name and password when starting the subagent. If the status is off, then the user name and password will be taken from the secure store. Default = Off.

Examples:

```
INTERACTIVE on
```

```
INTERACTIVE off
```

- `INTERACTION value`

  Where *value* is the number of interaction table entries. Range = 1 to 10. Default = 4.

  Examples:

  ```
  INTERACTION 4
  ```

  ```
  INTERACTION 2
  ```

- `MONITOR status`

  Where *status* is either on or off. Default = On.

  Examples:

  ```
  MONITOR on
  ```

  ```
  MONITOR off
  ```

- SSLKEY *certificate_file*

  Where *certificate_file* is the exported certificate along with the path. You must enter the path where this exported certificate exists.

  Examples:

  SSLKEY `/home/guest/snmp-cert.der` (Linux)

  SSLKEY `c:\home\guest\snmp-cert.der` (Windows)

NOTE: This option is not supported if there are multiple instances to be monitored that do not accept a common certificate.

- SERVER *hostname/IP_address:NCP_port*

  Where *hostname* is the name of the host where the OpenText eDirectory server is installed and configured. Only the locally installed server is supported.This is a required command in the file, otherwise none of the servers are monitored. Default: hostname of the local server.

  Examples:

  ```
  SERVER myserver
  ```

  ```
  SERVER myserver:1524
  ```

  On Linux, if you have multiple instances of OpenText eDirectory, you can include all the eDirectory servers you want to monitor as follows:

  ```
  SERVER myserver:1524
  ```

  ```
  SERVER myserver:2524
  ```

  ```
  SERVER myserver:6524
  ```

NOTE: No spaces are allowed before or after “`:`” as part of the server command.

Dynamic Configuration

Dynamic configuration can be done in either of the following ways, anytime after the Directory service is up and running.

Command Line

A trap configuration command line utility can be used to configure SNMP traps for OpenText eDirectory.

The command line configuration utility can be used to:

- Enable or disable traps
- Set the trap interval
- Enable or disable failure traps
- List the enabled, disabled or all traps

NOTE: For more details, see “Configuring Traps” on page 515.

OpenText Identity Console Application

Traps can also be configured using OpenText Identity Console. OpenText Identity Console is a browser-based tool used for administering, managing, and configuring OpenText eDirectory objects. OpenText Identity Console gives you the ability to assign specific tasks or responsibilities to users and to present the user with only the tools (with the accompanying rights) necessary to perform those sets of tasks.

1. In the OpenText Identity Console, click the **SNMP** tile.
2. Click the SNMP `Group` object you want to configure.
3. Specify the configurable parameters in the **General** and **Traps** drop down menus.
4. Click **Save**, then click **OK** to save the new configuration settings.

NOTE: For more information, see the Managing SNMP Group Objects.

Setting Up SNMP Services for OpenText eDirectory

This section describes setting up the SNMP services for OpenText eDirectory on the following platforms:

- “Windows” on page 498
- “Linux” on page 499

Setting up SNMP services for OpenText eDirectory requires the following steps:

1. Configuring the master agent
2. Starting the master agent
3. Configuring the subagent
4. Starting the subagent

Windows

- "Configuring the Master Agent" on page 498
- "Starting the Master Agent" on page 498
- "Stopping the Master Agent" on page 498
- "Starting the Subagent" on page 498

Configuring the Master Agent

NOTE: The SNMP master agent should be installed before OpenText eDirectory is installed. Refer to Microsoft SNMP Services (http://technet.microsoft.com/en-us/library/bb726977.aspx) for more details.

---

1. In the Microsoft SNMP Properties dialog box, click the **Agent** tab.
2. Enter the Contact and Location information.
3. Click the Traps, then enter the Community Name and Trap destination details.
   - **3a** Enter the Community Name, then click **Add**.
   - **3b** Enter the IP address or hostname of the destination computer that traps are generated for.
   - **3c** Click **Add** to add the IP address or hostname.
4. Enable the **Allow Service to Interact with Desktop** option.

   If this option is not enabled, you will be unable to connect to SNMP on Windows.

   On Windows platform: Click **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Services**. Then right-click **SNMP** and select **Properties**. At the **Log On** tab, select the **Allow Service to Interact with Desktop** option.

Starting the Master Agent

1. To start the master agent, do the following:

   Click **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Services** > **SNMP** > **Start**.
2. Enter the following at the command prompt:

   ```
   Net start SNMP
   ```

Stopping the Master Agent

To stop the master agent, do either of the following:

1. Click **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Services** > **SNMP** > **Stop**.
2. Enter the following at the command prompt:

   ```
   Net stop SNMP
   ```

Starting the Subagent

When the master agent starts on Windows, the subagent also starts.

IMPORTANT: The latest updated Service Pack needs to be installed after the installation of the SNMP service.

Linux

On Linux `net-snmp` should be installed. By default, it is installed on most Linux systems.

Setting up SNMP Services on Linux

- "Configuring the Master Agent" on page 499
- "Starting the Master Agent" on page 500
- "Starting the Subagent" on page 500
- "Stopping the Subagent" on page 500

Configuring the Master Agent

To configure the master agent on Linux, make the changes to your `snmpd.conf` file as mentioned in "snmpd.conf Changes" on page 499.

The `snmpd.conf` file is located in the `/etc/snmp` directory on SLES and in the `/etc` directory on other Linux platforms.

snmpd.conf Changes

In the `snmpd.conf` file, enter the following line:

```
trapsink myserver public
```

Where *myserver* is the hostname for the trap destination.

In the `snmpd.conf` file, add the following line:

```
master agentx
```

Additionally, make the following changes:

Original Content	Changed Content
`com2sec notConfigUser default public`	`com2sec demouser default public`
`group notConfigGroup v1 notConfigUser`	`group demogroup v1 demouser`
`view systemview included system`	view all included .1
`access notConfigGroup "" any noauth exact systemview none none`	`access demogroup "" any noauth exact all all all`

If the above content is not present in the `snmpd.conf` file, add it.

IMPORTANT: If any configuration files are changed, the master agent and subagent should be restarted.

Starting the Master Agent

To start the master agent, execute the following command:

```
/usr/sbin/snmpd -C -c /etc/snmpd.conf
```

NOTE: Run `/etc/init.d/snmpd start` command to start the master agent on SLES 12 and above.

Starting the Subagent

To start the subagent, execute the following command:

```
/etc/init.d/ndssnmpsa start
```

Enter the user name and password when prompted. Upon successful authentication, the following message is displayed if `INTERACTION = ON` in the `/etc/opt/novell/eDirectory/conf/ndssnmp/ndssnmp.cfg` file:

```
Do you want to remember password? (Y/N)
```

Enter Y to remember the password. When you start the subagent the next time, you are not prompted for the password.

Enter N to enter the password when the subagent is started the next time.

NOTE: When the server goes down, the master agent and subagent also go down. Therefore, to start the master agent and the sub-agent during server reboot time, execute the following commands:

```
chkconfig snmpd on
chkconfig ndssnmpsa on
```

Stopping the Subagent

To stop the subagent, execute the following command:

```
/etc/init.d/ndssnmpsa stop
```

Monitoring OpenText eDirectory Using SNMP

OpenText eDirectory is monitored using the traps and statistics feature of SNMP.

To monitor an OpenText eDirectory server using SNMP, you need the following rights over the NCP server, LDAP group and LDAP server objects:

- Supervisor rights over the NCP server object
- Read rights over the LDAP Allow Clear Text Password attribute of the LDAP Group object
- Read rights over the LDAP TCP Port and LDAP SSL Port attributes of the LDAP Server object

By default a user who has logged in with the administrative rights does not face any problem in monitoring an OpenText eDirectory server using SNMP.

Traps

The SNMP component generates a total of 119 traps out of which traps ndsServerStart (2001) and ndsServerStop (2002) cannot be configured. These traps are enabled by default.

You can use a MIB browser to check the generated traps.

Trap Number	Trap Name	Trap Is Generated When
1	ndsCreateEntry	A new object is added in the directory. Example: Create an object using LDAP tools, ICE, or OpenText Identity Console.
2	ndsDeleteEntry	An existing object is deleted. Example: Create an object using LDAP tools, ICE, or OpenText Identity Console.
3	ndsRenameEntry	An existing object is renamed. Example: Rename an object using LDAP tools, ICE, or OpenText Identity Console.
4	ndsMoveSourceEntry	An object is moved to a different context. The trap gives the context of the object before movement. Example: Move an object using ldapmodrdn or ldapsdk.
5	ndsAddValue	A value is added to an object attribute. Example: Add new values to attributes using LDAP tools, ICE, or OpenText Identity Console. **NOTE:** If the return value is NULL, you might have to access the directory over a secure channel. For more information, refer to "Accessing the Encrypted Attributes" on page 514

Trap Number	Trap Name	Trap Is Generated When
6	ndsDeleteValue	A value is deleted from an object attribute. Example: Delete new values to attributes using LDAP tools, ICE, or OpenText Identity Console. **NOTE:** If the return value is NULL, you might have to access the directory over a secure channel. For more information, refer to "Accessing the Encrypted Attributes" on page 514
7	ndsCloseStream	A stream attribute is modified.
8	ndsDeleteAttribute	A value is deleted from a single-value attribute. Example: Delete an attribute using LDAP tools, ICE, or OpenText Identity Console. **NOTE:** If the return value is NULL, you might have to access the directory over a secure channel. For more information, refer to "Accessing the Encrypted Attributes" on page 514.
9	ndsCheckSecurityEquiv	The security equivalence vector for the particular entry is checked. Example: Change the security equivalence attribute using LDAP tools, ICE, or OpenText Identity Console.
10	ndsUpdateSecurityEquiv	The security equivalence vector for the particular entry is modified. Example: Change the security equivalence attribute using LDAP tools, ICE, or OpenText Identity Console.
11	ndsMoveDestEntry	An object is moved to a different context. The trap will give the context that the object is moved to. Example: Move objects using ldapmodrdn or ldapsdk.
12	ndsDeleteUnusedExtref	A backlink object is deleted.
13	ndsAgentOpenLocal	The local directory agent is opened. Example: Run unattended repair.

Trap Number	Trap Name	Trap Is Generated When
14	ndsAgentCloseLocal	The local directory agent is closed. Example: Run unattended repair.
15	ndsDSABadVerb	An incorrect verb number is associated with an DSAgent request. Example: Pass a bad verb request to OpenText eDirectory using DClient calls.
16	ndsMoveSubtree	A container and its subordinate object are moved. Example: When a partition is moved to a different context using LDAP tools, ICE, or OpenText Identity Console.
17	ndsNoReplicaPointer	A replica has no replica pointer associated with it.
18	ndsSyncInEnd	Inbound synchronization is completed.
19	ndsBacklinkSecurEquiv	A backlink operation has updated an object's security equivalence vector. Example: Change the security equivalence attribute using LDAP tools, ICE, or OpenText Identity Console.
20	ndsBacklinkOperPrivChg	A backlink operation has changed an object's console operator privileges.
21	ndsDeleteSubtree	A container and its subordinate objects have been deleted.
22	ndsReferral	A referral is created.
23	ndsUpdateClassDef	A schema class definition is updated. Example: When a new class or attribute is added to a primary and this gets synchronized with the secondary using LDAP tools, ICE, or OpenText Identity Console, this trap is generated.
24	ndsUpdateAttributeDef	A schema attribute definition is updated. Example: When a new attribute is added to a primary and this is synchronized with the secondary using LDAP tools, ICE, or OpenText Identity Console, this trap is generated.
25	ndsLostEntry	OpenText eDirectory encounters a lost entry. A lost entry is an entry that does not exist on the local server, but for which updates are being received.
26	ndsPurgeEntryFail	The purge operation fails.

Trap Number	Trap Name	Trap Is Generated When
27	ndsPurgeStart	The purge operation is started. Example: Run DSTrace and `Set ndstrace=*j`.
28	ndsPurgeEnd	The purge operation is completed. Example: Run DSTrace and `Set ndstrace=*j`.
29	ndsLimberDone	The limber operation is completed. Example: Configure DSTrace to start limber after a particular interval of time.
30	ndsPartitionSplitDone	The split partition operation is completed. Example: Create a partition using OpenText Identity Console.
31	ndsSyncServerOutStart	Outbound synchronization from a particular server is started. Example: Configure DSTrace to start outbound synchronization after a particular interval of time.
32	ndsSyncServerOutEnd	Outbound synchronization from a particular server is completed. Example: Configure DSTrace to stop outbound synchronization after a particular interval of time.
33	ndsSyncPartitionStart	Partition synchronization is started. Example: Partition one of the containers.
34	ndsSyncPartitionEnd	Partition synchronization is completed. Example: Partition one of the containers.
35	ndsMoveTreeStart	Movement of a subtree is started. A subtree is moved when a partition is moved. Example: Using OpenText Identity Console, create a partition and move the partition to another container.

Trap Number	Trap Name	Trap Is Generated When
36	ndsMoveTreeEnd	Movement of a subtree is completed. A subtree is moved when a partition is merged. Example: Using OpenTextIdentity Console, create a partition and move the partition to another container.
37	ndsJoinPartitionDone	Joining of partitions is completed. Example: Using OpenText Identity Console, create a partition and merge the partition.
38	ndsPartitionLocked	A partition gets locked (for example, before merging the partitions). Example: Using OpenText Identity Console, create a partition.
39	ndsPartitionUnlocked	A partition gets unlocked (for example, after merging the partitions). Example: Using OpenText Identity Console, create a partition.
40	ndsSchemaSync	Schema are synchronized. Example: Schedule schema synchronization using `ldapsdk schsync`.
41	ndsNameCollision	Two objects on different servers have the same name (they **collide**). Example: Disable the outbound synchronization of the primary and secondary servers of a tree using iMonitor. Add some User objects to both the servers using LDAP tools. Then enable the outbound synchronization of both servers using iMonitor.
43	ndsChangeModuleState	An eDirectory module (NLM / DLM) is loaded or unloaded. Example: Load or unload the nldap module.
44	ndsLumberDone	The limber background process is started.
45	ndsBacklinkProcDone	The backlink process is completed. Example: Configure DSTrace to start backlink after a particular interval of time.

Trap Number	Trap Name	Trap Is Generated When
46	ndsServerRename	A server is renamed. Example: Use ldapmodrdn or ldapsdk to rename the server.
47	ndsSyntheticTime	Objects are created with future time stamps. To synchronize eDirectory servers, synthetic time might be invoked. Example: Add a secondary server to the tree using ndsconfig.
48	ndsServerAddressChange	Limber changes a server referral. Example: Change the IP address of the server and restart ndsd.
49	ndsDSARead	An entry is read. This trap is generated for all operations on OpenText eDirectory. Example: Use ldapsearch to generate traps.
50	ndsLogin	OpenText eDirectory is logged in to. Example: Login to the tree using ndslogin.
51	ndsChangePassword	A password is changed. Example: Change the password of a user object using ldapmodify.
52	ndsLogout	OpenText eDirectory is logged out of. Example: Detach the connection to the tree from Novell Client.
53	ndsAddReplica	A replica is added to a server partition. Example: Add a new replica to the tree using ndsconfig.
54	ndsRemoveReplica	A replica is deleted. Example: Delete a replica from one of the servers using OpenText Identity Console.

Trap Number	Trap Name	Trap Is Generated When
55	ndsSplitPartition	A partition is split. Example: Create a partition using OpenText Identity Console.
56	ndsJoinPartition	A parent partition is joined with a child partition. Example: Create a partition and join the partition using OpenText Identity Console.
57	ndsChangeReplicaType	A partition replica's type is changed. Example: Change the replica type from Master replica to Read-Write replica.
58	ndsAddEntry	A new object is added. Example: Add a user object using OpenText Identity Console.
59	ndsAbortPartitionOp	A partition operation is aborted. Example: Partition a container and abort the partitioning operation.
60	ndsRecvReplicaUpdates	A replica receives an update during synchronization. Example: An OpenText eDirectory server in a multiple-server tree setup requests updates on the replica that it holds. This operation can be done using OpenText Identity Console.
61	ndsRepairTimeStamps	A replica's time stamps are repaired. Example: Perform a DIB repair operation for timestamps using DSRepair (ndsrepair on Linux, or NDSCons on Windows).
62	ndsSendReplicaUpdates	A replica is updated during synchronization. Example: When an OpenText eDirectory server in a multiple servers tree setup sends for updates on the replica that it holds. This operation can be done using OpenText Identity Console.

Trap Number	Trap Name	Trap Is Generated When
63	ndsVerifyPass	A password is verified. Example: When the password expires, re-enter the password for confirmation at the change password prompt.
64	ndsBackupEntry	An entry is backed up. Example: Back up Directory objects using the Backup utility (ndsbackup on Linux, NDSCons on Windows).
65	ndsRestoreEntry	An entry is restored. Example: Restore the backed-up Directory objects using the Backup utility (ndsbackup on Linux , NDSCons on Windows).
66	ndsDefineAttributeDef	An attribute definition is added to the schema. Example: Extend the OpenText eDirectory tree schema by adding a new attribute definition. The schema can get extended when an OpenText eDirectory dependent application is installed such as ZENWorks® or NMAS™. The schema can also be extended using OpenText Identity Console or the schema extension utility ndssch on Linux.
67	ndsRemoveAttributeDef	An attribute definition is removed from the schema. Example: Delete an attribute definition from the OpenText eDirectory tree schema. The attribute can be deleted using OpenText Identity Console or the schema extension utility ndssch on Linux.
68	ndsRemoveClassDef	A class definition is removed from the schema. Example: Delete an object class definition from the OpenText eDirectory tree schema. This can be deleted using OpenTextIdentity Console or the schema extension utility ndssch on Linux.
69	ndsDefineClassDef	A class definition is added to the schema. Example: Extend the OpenText eDirectory tree schema by adding a new class. The schema can get extended when an OpenText eDirectory dependent application is installed such as ZENWorks or NMAS. The schema can also be extended using OpenText Identity Console or the schema extension utility ndssch on Linux.

Trap Number	Trap Name	Trap Is Generated When
70	ndsModifyClassDef	A class definition is modified. Example: Modify an existing object class or attribute definitions.
71	ndsResetDSCounters	The internal OpenText eDirectory counters are reset.
72	ndsRemoveEntryDir	A file directory associated with an entry is removed.
73	ndsCompAttributeValue	Attribute values are compared. Example: Compare an attribute value against any object.Perform an LDAP search operation against a User object to check if its telephone number is the same as the input value.
74	ndsOpenStream	A stream attribute is opened or closed. Example: Create or open a stream for read or write operations. Create a login script for a User object. It creates a file under the DIB directory, which results in the generation of this trap.
75	ndsListSubordinates	A List Subordinate Entries operation is performed on a container object. It is a one-level search. Example: Using OpenText Identity Console, click a container object to list the objects under it.
76	ndsListContainerClasses	A List Containable Classes operation is performed on an entry. Example: For a given object, list the container classes that can contain the given object. When queried against a user object, the container classes that can contain it are Organization, Organizational Unit, and Domain Classes.
77	ndsInspectEntry	An Inspect Entry operation is performed on an entry. Example: Inspect any entry to obtain information about the entry and to check if there are any errors that the entry has experienced.This event is generated as part of the Flat Cleaner background process of OpenText eDirectory, which results in this trap generation.

Trap Number	Trap Name	Trap Is Generated When
78	ndsResendEntry	A Resend Entry operation is performed on an entry. Example: During replication operation when an entry is resent because of a failure in sending the object earlier as a result of connection between the servers.
79	ndsMutateEntry	A Mutate Entry operation is performed on an entry. Example: Mutate a bindery object class to `User` object class.
80	ndsMergeEntries	Two entries are merged. Example: Merge two `User` objects. Merge `Entry2` (`ndsEntryName2`) into `Entry` (`ndsEntryName`).
81	ndsMergeTree	Two OpenText eDirectory trees are merged. Example: Merge two eDirectory trees using DSMerge (ndsmerge on Linux, NDSCons on Windows).
82	ndsCreateSubref	A subordinate reference is created. Example: Delete the replica of the child partition from a server, the Subordinate Reference replica gets created automatically which results in the generation of this trap.
83	ndsListPartitions	A List Partitions operation is performed. Example: Using OpenText Identity Console, from Partition and Schema view, click the OpenText eDirectory Server object to list the partitions held by the server.
84	ndsReadAttribute	A value of an attribute is read. Example: Perform a search operation on the tree.
85	ndsReadReferences	An entry's references are read.
86	ndsUpdateReplica	An Update Replica operation is performed on a partition replica. Example: Delete a user from one of the servers. The other replica is updated for the delete operation.

Trap Number	Trap Name	Trap Is Generated When
87	ndsStartUpdateReplica	A Start Update Replica operation is performed on a partition replica. Example: Delete a user from one of the servers. The other replica is updated for the delete operation.
88	ndsEndUpdateReplica	An End Update Replica operation is performed on a partition replica. Example: Delete a user from one of the servers. The other replica is updated for the delete operation.
89	ndsSyncPartition	A Synchronize Partition operation is performed on a partition replica. Example: Delete a user from one of the partitions. The sync can be observed using DSTrace.
90	ndsSyncSchema	The master replica of the root receives a request to synchronize its schema with the server. Example: Add a new class using OpenText Identity Console, LDAP tools, or ndssch utilities.
91	ndsCreateBackLink	A backlink is created. A backlink is created when an object not present locally is being referenced. Example: In a multi-server scenario, create a partition with some users. Delete this partition from one of the servers. This will create a subordinate reference. A backlink will be created for all the users present in the deleted partition.
93	ndsChangeTreeName	The tree name is changed. Example: Using the merge utility DSMerge/ndsmerge to rename the tree.
94	ndsStartJoinPartition	A Start Join operation is performed to merge partitions. Example: Merge or join partitions using LDAP tools.
95	ndsAbortJoinPartition	A Join Partition operation is aborted to stop merge partition. Example: Merge or join partitions using LDAP tools.

Trap Number	Trap Name	Trap Is Generated When
96	ndsUpdateSchema	An Update Schema operation is performed. Example: Add a new class using OpenText Identity Console, LDAP tools, or ndssch.
97	ndsStartUpdateSchema	A Start Update Schema operation is performed. Example: Add a new class using OpenText Identity Console, LDAP tools, or ndssch.
98	ndsEndUpdateSchema	An End Update Schema operation is performed. Example: Add a new class using OpenText Identity Console, LDAP tools, or ndssch.
99	ndsMoveTree	A Move Tree operation is performed. Example: Move a partition from one container to another.
101	ndsConnectToAddress	A connection is established with a particular address. Example: Browse the tree using OpenText Identity Console.
102	ndsSearch	A Search operation is performed. Example: Perform ldapsearch on the tree using LDAP tools.
103	ndsPartitionStateChange	A partition is created or deleted. Example: Create a new partition.
104	ndsRemoveBacklink	Unused external references are removed and the server sends a remove backlink request to the server holding the object.
105	ndsLowLevelJoinPartition	A low-level join is performed during merge partition operations. Example: Merge or join partitions using OpenText Identity Console or LDAP tools.
106	ndsCreateNameBase	An OpenText eDirectory namebase is created.

Trap Number	Trap Name	Trap Is Generated When
107	ndsChangeSecurityEquals	The Security Equals attribute is modified. Example: Change the security equivalent of any user and make it equal to `admin` using OpenText Identity Console.
108	ndsRemoveEntry	An entry is removed from OpenText eDirectory. Example: Delete any user using OpenText Identity Console.
109	ndsCRCFailure	A CRC failure occurs when fragmented NCP requests are being reconstructed.
110	ndsModifyEntry	An OpenText eDirectory entry is modified. Example: Modify attributes of any user using OpenText Identity Console.
111	ndsNewSchemaEpoch	The schema is reset using DSRepair. Example: Create a new schema epoch using `ndsrepair -S -Ad` on Linux.
112	ndsLowLevelSplitPartition	A low-level split is performed when a partition is being created. Example: Create a partition using OpenText Identity Console or LDAP tools.
113	ndsReplicaInTransition	A replica is added or removed.
114	ndsAclModify	A trustee of an object is changed (an Access Control List (ACL) object is changed). Example: Add, modify, or delete a trustee of an object using LDAP tools, ICE, or OpenText Identity Console.
115	ndsLoginEnable	A request for enabling the user account is received by the server. Example: Enable the Account Disable attribute using LDAP tools, ICE, or OpenText Identity Console.
116	ndsLoginDisable	A request for disabling the user account is received by the server. Example: Disable the Account Disable attribute using LDAP tools, ICE, or OpenText Identity Console.

Trap Number	Trap Name	Trap Is Generated When
117	ndsDetectIntruder	A user account is locked out because of intruder detection. Example: Locked by Intruder attribute using LDAP tools, ICE, or OpenText Identity Console.
2001	ndsServerStart	The subagent successfully reconnects to the OpenText eDirectory server. This trap consists of two variables: ◆ `ndsTrapTime`: This variable contains the total number of seconds since midnight (12 a.m.) of 1 January 1970 GMT (UT), when the subagent successfully reconnected to the OpenText eDirectory server. ◆ `ndsServerName`: OpenText eDirectory server to which the subagent reconnected successfully. Example: Bring down and bring up the OpenText eDirectory server when the subagent is up and running.
2002	ndsServerStop	The subagent loses its connection with the OpenText eDirectory server. This trap consists of two variables: ◆ `ndsTrapTime`: This variable contains the total number of seconds since midnight (12 a.m.) of 1 January 1970 GMT (UT), when the subagent lost connection with the eDirectory server. ◆ `ndsServerName`: eDirectory server to which the subagent lost its connection. Example: Bring down the OpenText eDirectory server when the subagent is up and running.

Accessing the Encrypted Attributes

OpenText eDirectory allows you to protect specific sensitive data when you store them on the disk and when you are trying to access them over the wire, by encrypting them. You can specify if you always need a secure channel to access the encrypted attributes or not. For more information, refer to “Accessing the Encrypted Attributes” on page 300.

When you have specified that you need only secure channels to access the encrypted attributes, NDS Value Events are blocked.Traps that are related to value events will have value data as NULL and you get an error, -6089, indicating that you need a secure channel to get the encrypted attributes value. Following are the traps which will have the value data as NULL:

- ndsAddValue
- ndsDeleteValue
- ndsDeleteAttribute

Configuring Traps

The method of configuring traps differs from platform to platform.

Platform	Utility
Windows	ndssnmpcfg
Linux	ndssnmpconfig

Windows

The utility to configure traps on Windows is ndssnmpcfg. This utility is present in the *install_path*\ directory. Use this utility to enable and disable traps, set a time interval for individual traps, set a default time interval, enable traps for failure operations, and list all traps.

Usage:

```
ndssnmpcfg -h [hostname[:port]] -p password -a userFDN -c command
```

Parameter	Description
-h	DNS host name or IP address
-p	userFDN password for authentication
-a	Fully Distinguished Name of a user having administrative rights
-c	Trap Commands (See “Windows Trap Commands” on page 515.)

Windows Trap Commands

Trap Commands	Description	Usage

DISABLE	Disabling a trap refers to the NMS not receiving traps although they are being generated.	To disable specific traps (for example, traps 10, 11, and 100):
		`ndssnmpcfg "DISABLE 10, 11, 100"`
		To disable all traps except 10, 11, and 100:
		`ndssnmpcfg "DISABLE ID != 10, 11, 100"`
		To disable all traps in the range 20 to 30:
		`ndssnmpcfg "DISABLE 20-29"`
		To disable all traps:
		`ndssnmpcfg "DISABLE ALL"`

Trap Commands	Description	Usage

ENABLE	Enabling a trap refers to the NMS receiving traps when they are generated.	`ndssnmpcfg "ENABLE trapSpec"`
		trapSpec can be any one of the following:
		To enable specific traps (for example, traps 10, 11, and 100):
		`ndssnmpcfg "ENABLE 10, 11, 100"`
		To enable all traps except 10, 11, and 100:
		`ndssnmpcfg "ENABLE ID != 10, 11, 100"`
		To enable all traps in the range 20 to 30:
		`ndssnmpcfg "ENABLE 20-29"`
		To enable all traps:
		`ndssnmpcfg "ENABLE ALL"`

INTERVAL	This utility is used to set and view the time interval.
	The time interval determines how many seconds to delay before sending duplicate traps.
	The time interval set should be between 0 and 2592000 seconds.
	If the time interval set is out of range, then the default time interval is considered.
	If the time interval is set to zero, all the traps are sent.
		To view the time interval:
		`ndssnmpcfg "213,240,79 INTERVAL"`
		To set the time interval between multiple traps (for example, to set the time interval between traps 12, 17, and 101 to 5):
		`ndssnmpcfg "12 17 101 INTERVAL 5"`
		To view the default time interval:
		`ndssnmpcfg "DEFAULT INTERVAL"`
		To set the default time interval:
		`ndssnmpcfg "DEFAULT INTERVAL=10"`

Trap Commands	Description	Usage

LIST	Use this utility to view lists of trap numbers that meet specified criteria.	`ndssnmpcfg LIST trapSpec`
		
		trapSpec is used to specify groups of trap numbers and can be any of the following keywords:
		
		ALL, ENABLED, DISABLED, FAILED, or a logical expression
		
		Examples:
		
		To list all enabled traps along with trap names:
		
		`ndssnmpcfg LIST ENABLED`
		
		To list all disabled traps along with trap names:
		
		ndssnmpcfg LIST DISABLED
		
		To list all traps (117) along with trap names:
		
		`ndssnmpcfg LIST ALL`
		
		To list specific traps like 12, 224, and 300 along with trap names:
		
		`ndssnmpcfg LIST ID = 12,224,300`
		
		To list all traps except selected traps like 12, 224, and 300 along with trap names:
		
		`ndssnmpcfg LIST ID != 12,224,300`
		
		To list all traps which have been enabled for failure with trap names:
		
		`ndssnmpcfg LIST FAILED`

Trap Commands	Description	Usage
READ_CFG	Use this command to reconfigure the directory configuration from the configuration file `ndstrap.cfg`. Any changes specified in the configuration file will then take effect. This utility is primarily used to put various commands together in the `ndstrap.cfg` and do the operation in one instance. The `ndstrap.cfg` is located in *`install directory`*\SNMP The `ndstrap.cfg` file specifies operational parameters to be used for trap configuration and provides a way to configure the operation of SNMP traps. This file is read whenever the trap configuration utility, ndssnmpcfg is executed with the READ_CFG command.	`ndssnmpcfg "READ_CFG"`

FAILURE	This command is used to list all traps enabled for failure.
	
	Whenever an event fails, a failure trap is generated.
	
	NOTE: If the trap is enabled for failure and then disabled and again enabled using the enable trapid command, the trap is enabled for success and not for failure.
		`ndssnmpcfg "FAILURE` *`trapSpec`*`"`
		
		`trapSpec` consists of one or more trap numbers separated by commas or spaces, the keyword ALL, or a logical expression.Examples:
		
		To set failure for multiple traps:
		
		`ndssnmpcfg "FAILURE 10,11,100"`
		
		To set failure for all traps except the traps mentioned:
		
		`ndssnmpcfg "FAILURE ID != 24,30"`
		
		To set failure for all traps:
		
		`ndssnmpcfg "FAILURE ALL"`

Linux

The utility to configure traps on Linux is ndssnmpconfig. This utility is present in the `/etc/ndssnmp/` directory. Use this utility to enable and disable traps, set a time interval for individual traps, set a default time interval, enable traps for failure operations, and list all traps.

Usage:

`ndssnmpconfig -h [`*`hostname`*`[`*`:port`*`]] -p `*`password`*` -a `*`userFDN`*` -c `*`command`*

Parameter	Description
-h	DNS host name or IP address
-p	userFDN password for authentication
-a	Fully distinguished name of a user having administrative rights
-c	Trap commands (See “Linux Trap Commands” on page 519.)

Linux Trap Commands

Trap Commands	Description	Usage

DISABLE	Disabling a trap refers to the NMS not receiving traps though they are being generated.	To disable specific traps (for example, traps 10, 11 and 100):
		`ndssnmpconfig "DISABLE 10, 11, 100"`
		To disable all traps except 10, 11, and 100:
		`ndssnmpconfig "DISABLE ID != 10, 11, 100"`
		To disable all traps in the range 20 to 30:
		`ndssnmpconfig "DISABLE 20-29"`
		To disable all traps:
		`ndssnmpconfig "DISABLE ALL"`

Trap Commands	Description	Usage

ENABLE	Enabling a trap refers to the NMS receiving traps when they are generated.	`ndssnmpconfig "ENABLE trapSpec"`
		trapSpec can be any one of the following:
		To enable specific traps (for example, traps 10, 11, and 100):
		`ndssnmpconfig "ENABLE 10, 11, 100"`
		To enable all traps except 10, 11, and 100:
		`ndssnmpconfig "ENABLE ID != 10, 11, 100"`
		To enable all traps in the range 20 to 30:
		`ndssnmpconfig "ENABLE 20-29"`
		To enable all traps:
		`ndssnmpconfig "ENABLE ALL"`

INTERVAL	This utility is used to set and view the time interval.
	The time interval determines how many seconds to delay before sending duplicate traps.
	The time interval should be between 0 and 2592000 seconds.
	If the time interval is out of range, then the default time interval is considered.
	If the time interval is set to zero, all the traps are sent.
		To view the time interval:
		`ndssnmpconfig "213,240,79 INTERVAL"`
		To set the time interval between multiple traps (for example, to set the time interval between traps 12, 17, and 101 to 5):
		`ndssnmpconfig "12 17 101 INTERVAL 5"`
		To view the default time interval:
		`ndssnmpconfig "DEFAULT INTERVAL"`
		To set the default time interval:
		`ndssnmpconfig "DEFAULT INTERVAL=10"`

Trap Commands	Description	Usage

LIST	Use this utility to view lists of trap numbers that meet specified criteria.	ndssnmpconfig LIST <*trapSpec*>
		
		trapSpec is used to specify groups of trap numbers and can be any of the following keywords:
		
		ALL, ENABLED, DISABLED, FAILED, or a logical expression
		
		Examples:
		
		To list all enabled traps along with trap names:
		
		`ndssnmpconfig LIST ENABLED`
		
		To list all disabled traps along with trap names:
		
		`ndssnmpconfig LIST DISABLED`
		
		To list all traps (117) along with trap names:
		
		`ndssnmpconfig LIST ALL`
		
		To list specific traps like 12, 224, and 300 along with trap names:
		
		`ndssnmpconfig LIST ID = 12,224,300`
		
		To list all traps except selected traps like 12, 224, and 300 along with trap names:
		
		`ndssnmpconfig LIST ID != 12,224,300`
		
		To list all traps that have been enabled for failure with trap names:
		
		`ndssnmpconfig LIST FAILED`

Trap Commands	Description	Usage
READ_CFG	Use this command to reconfigure the directory configuration from the configuration file `ndstrap.cfg`. Any changes specified in the configuration file will then take effect. This utility is primarily used to put various commands together in the `ndstrap.cfg` file and perform the operation in one instance. The `ndstrap.cfg` file is located in `/etc/ndssnmp/`. The `ndstrap.cfg` file specifies operational parameters to be used for trap configuration and provides a way to configure the operation of SNMP traps. This file is read whenever the trap configuration utility ndssnmpcfg is executed with the READ_CFG command.	`ndssnmpconfig "READ_CFG"`

FAILURE	This command is used to list all traps enabled for failure.
	
	Whenever an event fails, a failure trap is generated.
	
	NOTE: If the trap is enabled for failure and then disabled and again enabled using the enable trapid command, the trap is enabled for success and not for failure.
		`ndssnmpconfig "FAILURE` *`trapSpec`*`"`
		
		`trapSpec` consists of one or more trap numbers separated by commas or spaces, the keyword ALL, or a logical expression.
		
		Examples:
		
		To set failure for multiple traps:
		
		`ndssnmpconfig "FAILURE 10,11,100"`
		
		To set failure for all traps except the traps mentioned:
		
		`ndssnmpconfig "FAILURE ID != 24,30"`
		
		To set failure for all traps:
		
		`ndssnmpconfig "FAILURE ALL"`

Statistics

- “ndsDbCache” on page 523
- “ndsDbConfig” on page 523
- “ndsProtoIfOps” on page 524
- “ndsServerInt” on page 526

ndsDbCache

Managed Objects in Directory	Description
ndsDbSrvApplIndex	An index to uniquely identify the OpenText eDirectory Server Application.
ndsDbDibSize	Current size of the OpenText eDirectory Database in KB.
ndsDbBlockSize	Block size of the OpenText eDirectory Database in KB.
ndsDbEntryCacheMaxSize	Information on max size of the entry cache in KB.
ndsDbBlockCacheMaxSize	Information on max size of the block cache in KB.
ndsDbEntryCacheCurrentSize	Information on the current entry cache size.
ndsDbBlockCacheCurrentSize	Information on the current block cache size.
ndsDbEntryCacheCount	Information on the number of entries in the cache.
ndsDbBlockCacheCount	Information on the number of blocks in the cache.
ndsDbEntryCacheOldVerCount	Information on prior version entries in the cache.
ndsDbBlockCacheOldVerCount	Information on prior version blocks in the cache.
ndsDbEntryCacheOldVerSize	Information on prior version entry cache size.
ndsDbBlockCacheOldVerSize	Information on prior version block cache size.
ndsDbEntryCacheHits	Information on the number of entry hits.
ndsDbBlockCacheHits	Information on the number of block hits.
ndsDbEntryCacheHitLooks	Information on the number of entries examined to find hits.
ndsDbBlockCacheHitLooks	Information on the number of blocks examined to find hits.
ndsDbEntryCacheFaults	Information on the number of entry faults.
ndsDbBlockCacheFaults	Information on the number of block faults.
ndsDbEntryCacheFaultLooks	Information on the number of entries examined to determine misses.
ndsDbBlockCacheFaultLooks	Information on the number of blocks examined to determine misses.

ndsDbConfig

Managed Objects in Directory	Description
ndsDbCfgSrvApplIndex	An index to uniquely identify the OpenText eDirectory Server Application.

Managed Objects in Directory	Description
ndsDbCfgDynamicCacheAdjust	Information on whether Dynamic Cache Adjust is on or off. 0 = off 1 = on
ndsDbCfgDynamicCacheAdjustPercent	Information on the Dynamic Cache Adjust percentage parameter of available memory.
ndsDbCfgDynamicCacheAdjustMin	Information on the Dynamic Cache Adjust Minimum value parameter. This is cache size constraint values in KB.
ndsDbCfgDynamicCacheAdjustMinToLeave	Information on the Dynamic Cache Adjust Minimum value parameter in KB that is to be subtracted from the total available memory in KB.
ndsDbCfgHardLimitCacheAdjust	Information on whether Hard Limit Cache Adjust is on or off. 0 = off 1 = on
ndsDbCfgHardLimitCacheAdjustMax	Information on the cache maximum size in KB. This is a hard limit parameter.
ndsDbCfgBlockCachePercent	Information on the block cache percentage.
ndsDbCfgCacheAdjustInterval	Information on the cache adjust interval in seconds.
ndsDbCfgCacheCleanupInterval	Information on the cache cleanup interval in seconds.
ndsDbCfgPermanentSettings	Information on whether Permanent Settings is on or off. 0 = off 1 = on

ndsProtoIfOps

Managed Objects in Directory	Description
ndsProtoIfSrvApplIndex	An index to uniquely identify the OpenText eDirectory Server Application.
ndsProtoIfIndex	An index to uniquely identify an entry corresponding to an OpenText eDirectory Server protocol interface.
ndsProtoIfDescription	Information on the port being used by the DS protocol interface.
ndsProtoIfUnauthBinds	Number of unauthenticated/anonymous bind requests received.
ndsProtoIfSimpleAuthBinds	Number of bind requests that were authenticated using simple authentication procedures where the password is sent over the wire in encrypted or clear text format.
ndsProtoIfStrongAuthBinds	Number of bind requests that were authenticated using SASL and X.500 strong authentication procedures. This includes the binds that were authenticated using external authentication procedures.

Managed Objects in Directory	Description
ndsProtoIfBindSecurityErrors	Number of bind requests that have been rejected due to inappropriate authentication or invalid credentials.
ndsProtoIfInOps	Number of requests received from DUAs or other OpenText eDirectory servers.
ndsProtoIfReadOps	Number of read requests received.
ndsProtoIfCompareOps	Number of compare requests received.
ndsProtoIfAddEntryOps	Number of addEntry requests received.
ndsProtoIfRemoveEntryOps	Number of removeEntry requests received.
ndsProtoIfModifyEntryOps	Number of modifyEntry requests received.
ndsProtoIfModifyRDNOps	Number of modifyRDN requests received.
ndsProtoIfListOps	Number of list requests received.
ndsProtoIfSearchOps	Number of search requests (baseObject searches, oneLevel searches, and whole subtree searches) received.
ndsProtoIfOneLevelSearchOps	Number of oneLevel search requests received.
ndsProtoIfWholeSubtreeSearchOps	Number of whole subtree search requests received.
ndsProtoIfExtendedOps	Number of extended operations.
ndsProtoIfReferrals	Number of referrals returned in response to requests for operations.
ndsProtoIfChainings	Number of operations forwarded by this OpenText eDirectory server to other OpenText eDirectory servers.
ndsProtoIfSecurityErrors	Number of requests received that did not meet the security requirements.
ndsProtoIfErrors	Number of requests that could not be serviced because of errors other than security errors and referrals. A partially serviced operation is not counted as an error. The errors include naming-related, update-related, attribute-related, and service-related errors.
ndsProtoIfReplicationUpdatesIn	Number of replication updates fetched or received from OpenText eDirectory servers.
ndsProtoIfReplicationUpdatesOut	Number of replication updates sent to or taken by OpenText eDirectory servers.
ndsProtoIfInBytes	Incoming traffic, in bytes, on the interface. This includes requests from DUAs as well as responses from other OpenText eDirectory servers.
ndsProtoIfOutBytes	Outgoing traffic, in bytes, on the interface. This includes responses to DUAs and OpenText eDirectory servers as well as requests to other OpenText eDirectory servers.

ndsServerInt

Managed Objects in Directory	Description
ndsSrvIntSrvApplIndex	An index to uniquely identify an OpenText eDirectory server application.
ndsSrvIntProtoIfIndex	An index to uniquely identify an entry corresponding to an OpenText eDirectory server protocol interface.
ndsSrvIntIndex	Together with ndsSrvIntSrvApplIndex and ndsSrvIntProtoIfIndex, this object forms the unique key to identify the conceptual row that contains useful information on the (attempted) interaction between the OpenText eDirectory server (referred to by applIndex) and a peer OpenText eDirectory server using a particular protocol.
ndsSrvIntURL	URL of the peer OpenText eDirectory server.
ndsSrvIntTimeOfCreation	The total number of seconds since midnight (12 a.m.) of 1 January 1970 GMT (UT) when this row was created.
ndsSrvIntTimeOfLastAttempt	The total number of seconds since midnight (12 a.m.) of 1 January 1970 GMT (UT) when the last attempt was made to contact the peer OpenText eDirectory server.
ndsSrvIntTimeOfLastSuccess	The total number of seconds since midnight (12 a.m.) of 1 January 1970 GMT (UT) when the last attempt made to contact the peer OpenText eDirectory server was successful.
ndsSrvIntFailuresSinceLastSuccess	The number of failures since the last time an attempt to contact the peer OpenText eDirectory server was successful. If there have been no successful attempts, this counter will contain the number of failures since this entry was created.
ndsSrvIntFailures	Cumulative failures in contacting the peer OpenText eDirectory server since the creation of this entry.
ndsSrvIntSuccesses	Cumulative successes in contacting the peer OpenText eDirectory server since the creation of this entry.

Troubleshooting

Log files are maintained to troubleshoot the problems that occur. These log files contain information about the errors that occur and can help you solve the problems. See “Troubleshooting SNMP” on page 811 for more details.

Table 18-1 lists the default location of the server log file for Linux platforms. To know the location of the `ndsd.log` file, run the `ndsconfig get n4u.server.log-file` command against the OpenText eDirectory instance.

Table 18-1 *Log File Location*

Platform	Subagent	Server	Master
Windows	*install_directory*\nds\snmp\dssnmpsa.log	*install_directory*\nds\snmp\dssnmpsrv.log	NA
Linux	/var/opt/novell/eDirectory/log/ndssnmpsa.log	/var/opt/novell/eDirectory/log/ndsd.log	/var/log/messages

19 Maintaining OpenText eDirectory

For OpenText eDirectory to perform optimally, you need to maintain the directory through routine health check procedures and upgrading or replacing hardware when necessary.

This chapter covers the following maintenance topics:

Performance

- “Advanced Referral Costing” on page 529

Health Checks

- “Keeping OpenText eDirectory Healthy” on page 538
- “Resources for Monitoring” on page 541

Hardware Replacements

- “Upgrading Hardware or Replacing a Server” on page 541

OpenText eDirectory Recovery

- “Restoring OpenText eDirectory after a Hardware Failure” on page 547

Advanced Referral Costing

Server applications often communicate with other servers via a built-in client (Dclient), because a single server doesn't contain all the necessary OpenText eDirectory data for an application to operate. An example is NLDAP, when it is configured to chain requests.

When a server application requests data that the local server does not hold, the server locates another server that contains the requested data, and subsequently retrieves the data for the client. This process is called “tree walking”. It naturally takes longer for a server to fulfill a request through tree walking. Although best practice guidelines for OpenText eDirectory tree design minimize the need for tree walking, it is still sometimes necessary.

Figure 19-1 *Advanced Referral Costing*

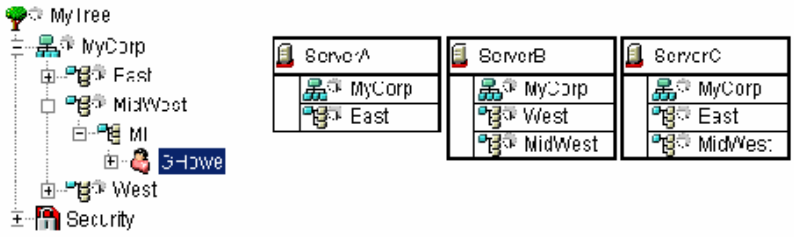


Figure 19-1 illustrates an LDAP subtree search to Server A for `cn=GHowe`, starting at `O=MyCorp`. However, the `cn=GHowe` object is located in the `ou=MidWest` partition, which is not represented on Server A.

To locate a server that holds the data needed to fulfill the client request, Server A must either get the data from Server B or Server C. To do this, Server A must send the request to either Server B or C. Server A happens to choose Server B. Note that the process of choosing server is unpredictable. Server B is available on the network and accepts the request, but is unable to complete the request quickly, resulting in Server A waiting for Server B even though Server C could also provide the required data. Until Server B either fulfills the request or is no longer available on the network, the request from Server A must wait.

The following sections provide information about how you can improve the performance of OpenText eDirectory servers:

- "Improving Server-to-Server Connection" on page 530
- "Advantages of Referral Costing" on page 532
- "Deploying ARC" on page 533
- "Enabling Advanced Referral Costing" on page 534
- "Tuning Advanced Referral Costing" on page 534
- "Monitoring Advanced Referral Costing" on page 535

Improving Server-to-Server Connection

Advanced Referral Costing (ARC) is an improved costing algorithm. The main purpose of ARC is to prevent server outages. Some of the benefits of ARC can include:

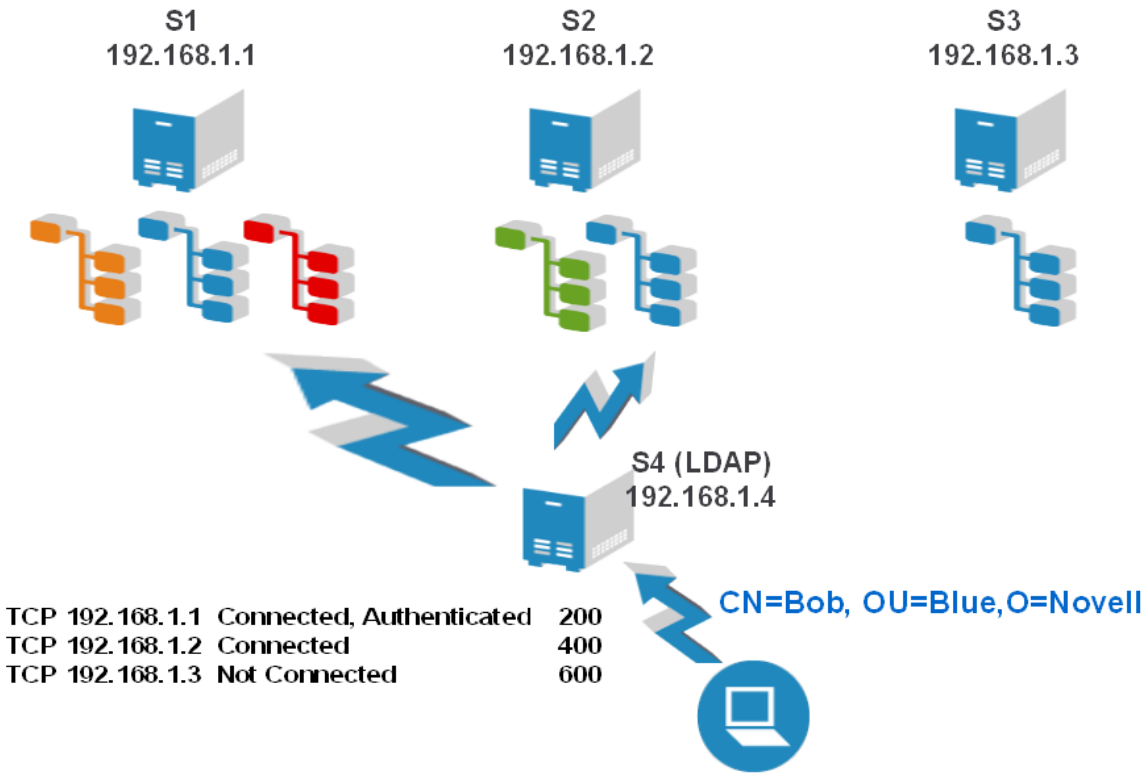
- Improved server performance and fault tolerance
- Better server-to-server communications
- Load distribution
- Remote server health monitoring
- Simplified isolation and identification of communication problems

Who Should Use ARC?

Servers that don't hold a local copy of an object or service need to walk the tree for information benefit from ARC, because they frequently communicate with the other servers. ARC is very effective in an LDAP environment, especially during prefer chaining.

For example, a server is sometimes overwhelmed by other servers that always make requests to that server, as illustrated in Figure 19-2.

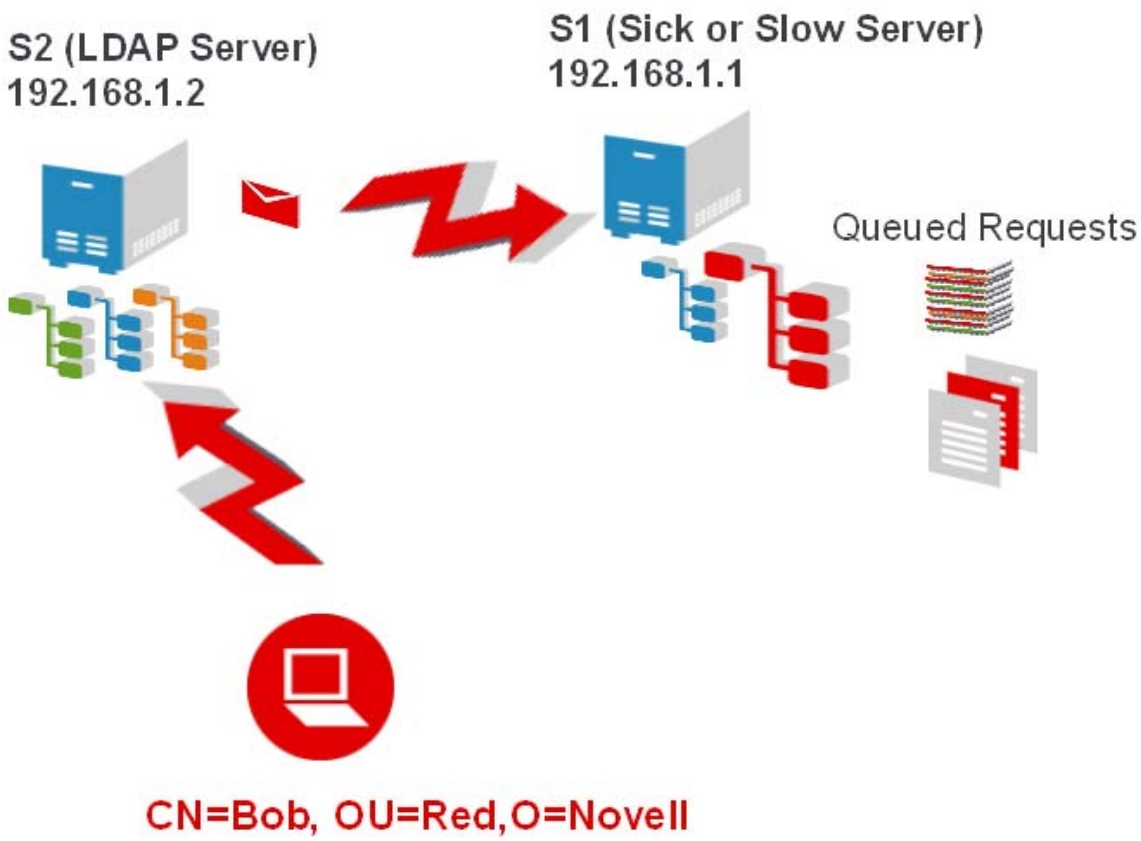
Figure 19-2 *One Stop Server Effect*



Although there are other available servers with replicas of the needed objects, servers still seem to prefer this server. This is because the servers making requests for a service or replica are are already connected to this server, so they tend to send all the requests that the server can handle. Figure 19-2 shows that all requests from S4 are going to S1. This is because S4 was already connected and authenticated to S1, so it continues to send all the requests for the blue partition to S1, even though S2 and S3 could service those requests. ARC helps to eliminate these situations by distributing the load to the servers that respond faster. You should enable ARC on remote servers (S4) that request this server, or you can enable ARC on all servers.

Figure 19-3 shows another scenario, illustrating the “cascading server” effect. Here, server S1 is often not responding, but it is not down. If the S1 were down, the requests would time out and communication would stop. If the server is still up at the transport level, but the database is slow or busy, the server continues to accept and queue new requests from other servers. This can cause the additional servers (S2) to eventually run out of threads. Each outstanding request takes a thread on the remote server, and when they run out of threads the server becomes non-responsive. ARC resolves this issue by distributing requests across the fastest servers, because a server that is slow or sick incurs a higher cost in servicing requests.

Figure 19-3 *Cascading Server Effect*



In addition, ARC is a good choice for improving fault tolerance. It has the ability to easily identify server communication problems.

Advantages of Referral Costing

- It times/routes most Resolve Name requests to remote servers as they are made.
- It averages the Resolve Name request times in milliseconds on each address. This allows ARC to be more granular and adjust the cost of the referral more aggressively. It is also able to quickly detect a slow server, because timing is tracked in milliseconds instead of seconds.
- It tracks outstanding requests so quickly determine if a request is taking too long. It does not have to wait for the request to complete in order to know that the server is taking a long time.
- It tracks response time on a per-address basis. It is normal for a server to have numerous connections to the same address. By tracking per address instead of per connection, one connection can benefit from statistics gathered from the other connections.

NOTE: To account for LDAP requests, ARC also takes into account responsiveness of private connections.

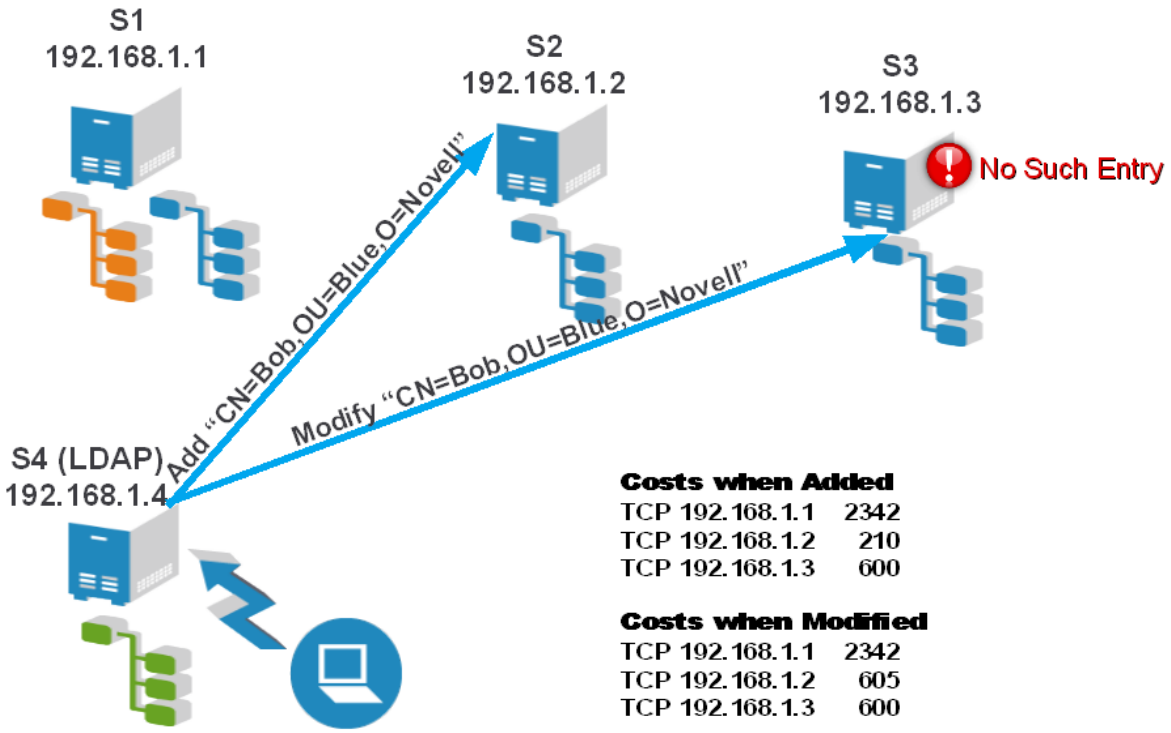
Deploying ARC

ARC is usually deployed on a server-to-server basis. Those servers that are ARC - enabled can know the new costing information. You should enable ARC on each server in the environment.

Deployment Considerations

It is not useful to enable ARC on all servers. Figure 19-4 shows a situation that could impact the efficiency of LDAP servers. In the figure, S4 holds a copy of the green partition, but not of the blue partition. Any chaining LDAP request that requires information from the blue partition needs to walk to the S1, S2, or S3 servers to be fulfilled. This works in most cases, and ARC is designed for just such situations.

Figure 19-4 *ARC Deployment Considerations*



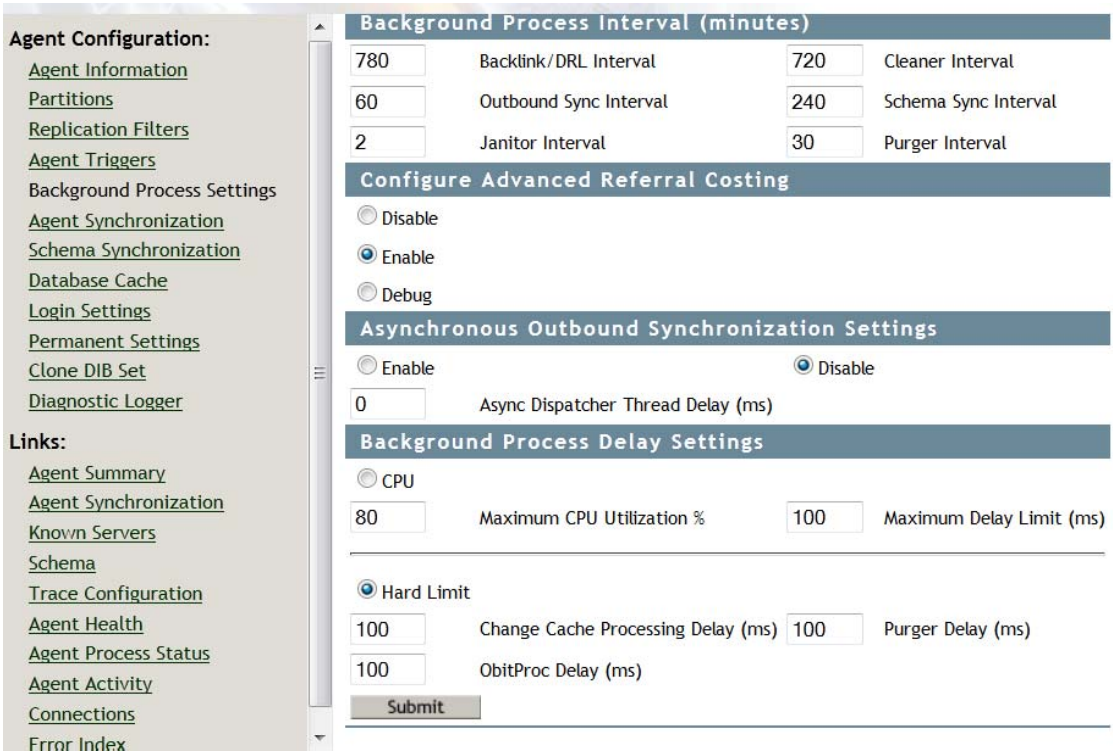
However, performing specific LDAP operations could be difficult. Although it is possible to add a user, for example, Bob.Blue.Novell, the operation might fail when you try to immediately return to modify Bob. The figure shows Bob added on S2, but modifying Bob on S3 has failed because S3 has not yet synchronized with S2, so S3 has not yet received Bob. ARC has the capability to direct you to a different server, because ARC is more dynamic than the original costing method.

This configuration works well in scenarios where the server costs don't vary much and they don't have problems synchronizing. Disabling ARC on S4 resolves this issue.

Enabling Advanced Referral Costing

ARC is enabled by default for OpenText eDirectory. To configure ARC by using the NDS OpenText iMonitor, click **Agent Configuration** > **Background Process Settings**. In addition, the **Enable**, **Disable**, and **Debug** options are available.

Figure 19-5 *NDS OpenText iMonitor Agent Configuration Screen*



NDSTrace

Use the NDSTrace tool to enable ARC on all UNIX platforms.

Table 19-1 *Enabling ARC on UNIX Platform*

`set NDSTRACE =!ARC`	Displays the gv_ResolveTimesTable for debugging.
`set NDSTRACE =!ARC0`	Disables Advanced Referral Costing.
`set NDSTRACE =!ARC1`	Enables Advanced Referral Costing.
`set NDSTRACE =!ARC2`	Enables Advanced Referral Costing in debug mode and displays the resulting costs of each referral on the Resolve Name DSTrace flag anytime a costing decision is made.

Tuning Advanced Referral Costing

ARC requires no tuning by default. However, there are tunable parameters in ARC that can be used to change how ARC functions, or to disable or enable certain features. There are 3 major components to ARC.

Advanced Costing

When asked to cost a given address, ARC uses the information known about the connection to calculate the cost of the given referral. If ARC is on, Advanced Costing is always used when costing a referral.

Background Monitoring

A background thread periodically checks the timer information to ensure that it is current. When a server is slow, its cost rises and there is a good chance that communication will cease. The background thread periodically (once a minute by default) checks to see if a server in the table has not been updated. If the server is has not been updated in the last three minutes, the server makes a resolve name request on its behalf to check the server's health. This creates current costing for the server, and also detects if a server is now less busy, or is healthy, so a client doesn't need to suffer adverse effects to check the server's health. There are two permanent configuration parameters that can be changed for the background thread:

- **ARC_MAX_WAIT:** How stale a timer is before a request to the server to check its health (180 seconds by default).
- **ARC_BG_INTERVAL:** How often the background thread runs (60 seconds by default,: 0 means disabled and the thread doesn't run).

For additional information, see section 8.4.24 setting permanent configuration parameters.

Remote Health Information

Servers using ARC periodically request health information from a remote server. These are not additional requests on the wire, but additional health information that is returned in standard resolve name requests that servers frequently make. This information is then used in the costing algorithm to further enhance reactions to servers that are under heavy loads. When a resolve name request is being made to a remote server, if it has been more than 15 seconds since the last update, health information is requested from the remote server and is added to the reply of the resolve name request.

There is one tunable parameter for Remote Health Monitoring:

- **ARC_DS_INFO_INTERVAL:** This is how often to request lock (health) information in ARC (15 seconds by default).

Monitoring Advanced Referral Costing

You can print the ResolveTimes table to observe Advanced Referral Costing in action.

Use the following commands to print the ResolveTimes table:

- `set DSTRACE = +DBG`
- `set DSTRACE = !ARC`

This prints the Resolve Times table and the current stored information for each server. It shows the transport address, the milliseconds since the address was last used, the last cost that was used in a referral decision, and the number of outstanding requests for that address.

A high number of outstanding requests is not necessarily a problem. It might simply mean that server is used frequently.

Using ARC for Troubleshooting

One of the most useful features of ARC is the ability to quickly identify communication problems with servers.

The following is an example of a ResolveTimesTable printout:

ARC is currently enabled.

Table 19-2 *Resolve Time Costs*

Slot	Transport Address	Cost	LastUse	Checked	#Req	waiters	LockTime
1	tcp:151.155.134.27:524	214	14	14	0	0	0
2	tcp:151.155.134.11:524	0	0	0	0	0	0
3	udp:151.155.134.11:524	0	0	0	0	0	0
4	cp:151.155.134.13:524	554759	280	0	0	27	582
5	tcp:151.155.134.59:524	0	179	179	0	0	0
6	udp:151.155.134.59:524	0	119	119	0	0	0
7	tcp:151.155.134.28:524	1543	119	119	0	0	0
8	tcp:151.155.134.15:524	124	14	14	0	0	0

The printout shows that from this server's perspective, 151.155.134.13 is having difficulties. You can also see that the problem is most likely the server, not the transport. The server has 27 requests waiting for access to the database, and the requests are taking a long time to acquire the database lock. This server has two requests that have never received replies from the remote server.

You can also see that 151.155.134.11 and 151.155.134.59 are either very fast servers, or are not very busy, or both. You can see that 151.155.134.59 and 151.155.134.11 have both had problems communicating via TCP at one time, but are both healthy now, because they both have UDP connections. UDP connections to a server are tried only if there is a problem talking to the server via TCP.

The following is a summary of what each number means:

Transport Address: The address of the remote server.

Cost: The current cost of the remote server.

Last Use: The duration in seconds since last communication with the server.

Checked: The duration in seconds since last health information from the remote server.

#Req: The number of outstanding requests to the remote server.

Waiters: The number of requests to the remote server waiting for the database lock.

LockTime: Duration that a process has held the database lock on the remote server.

The following printout has another example of quickly identifying a communications problem, because you can see that the server currently cannot communicate to 151.155.134.13 via TCP.

ARC is currently enabled.

Table 19-3 *Resolve Time Costs*

Slot	Transport Address	Cost	LastUse	Checked	#Req	waiters	LockTime
1	tcp:151.155.134.27:524	394	92	14	0	0	0
2	tcp:151.155.134.11:524	0	0	0	0	0	0
3	udp:151.155.134.11:524	0	0	0	0	0	0
4	tcp:151.155.134.13:524	5000000	180	180 is in BAD ADDRESS CACHE			

There are a few things to keep in mind when looking at these tables:

- Outstanding requests are not necessarily bad, because the server might just be servicing many requests. Outstanding requests on servers where costing is high are a problem.
- Your first indicator of a server's health is the current cost, making it easy to see what server is causing you problems.

NOTE: All requests are timing round trip time, and how long requests are outstanding. This means transport times are also a component of the cost. If a server shows up as having problems in this table, but is working well from other servers, and doesn't appear to have a problem, this might indicate a transport issue.

Background Thread Traces

The following is a trace showing the ARCBackgroundResolveTimerThread running:

```
ARCBackGroundResolveTimerThread started Interval = 60 MaxWait = 180000

Updating timer info for tcp:151.155.134.11:524

Updating timer info for udp:151.155.134.11:524

Updating timer info for tcp:151.155.134.13:524
ARCBackGroundResolveTimerThread error -635 in DCConnectToAddress for
tcp:151.155.134.59:524

ARCBackGroundResolveTimerThread completed in 0 seconds

8-total timers 4-stale timers 3-timers updated
```

From the above message you can see the following:

- TCP:151.155.134.11 has not been used for more than 3 minutes
- UDP:151.155.134.11 has not been used for more than 3 minutes
- TCP: 151.155.134.13 has not been used for more than 3 minutes

The timer information was updated for all of the above servers, with the following results:

- TCP: 151.155.134.59 is still not reachable from this server.

The new costing is very dynamic and changes very frequently. In order to watch it work, you can set the Advanced Referral Costing parameter to Debug mode.

NOTE: Ensure you reset ARC to non debug mode by running the command `set NDSTRACE = !ARC1` when you have finished monitoring. Overhead printing costs are not desirable when you don't need it.

In the DSTrace or NDSTrace, you now see the individual referral costs displayed if Advanced Referral Costing and +RSLV are turned on. The remaining tags are turned off using the `set NDSTrace =nodebug` command.

```
Sorted results from DCAdjustCostAndSort follow:

137.65.10.3 cost of 217

137.65.10.9 cost of 222

137.65.10.10 cost of 400
```

The numbers change quickly if a remote server is slow or overloaded. The ExRef server's costing adjusts dynamically every second, so to watch costs over time you should the trace to a log file.

Keeping OpenText eDirectory Healthy

The health of directory services is vital to any organization. Regular health checks using OpenText iMonitor will keep your directory running smoothly and will make upgrades and troubleshooting much easier.

When to Perform Health Checks

In general, if your network doesn't change often (servers and partitions are added only every couple of months and only simple changes are made frequently), perform health checks once a month.

If your network is more dynamic (partitions or servers are added weekly or your organization is reorganizing), perform health checks weekly.

Adjust the frequency of health checks as your environment changes. Factors that influence the timing of your health checks include the following:

- Number of partitions and replicas
- Stability of replica holding servers
- Amount of information in an OpenText eDirectory partition
- Object size and complexity
- Number of errors in previous DSRepairs

When you perform a health check, OpenText iMonitor gathers information from all servers based on given rights. Be aware that running health check reports might generate network traffic and use disk space.

Health Check Overview

A complete health check includes checking the following:

- OpenText eDirectory version

  Running different versions of NDS or OpenText eDirectory on the same server can cause synchronization problems. If your version of NDS or OpenText eDirectory is outdated, download the latest software patch from the Software License and Download portal.

- Time synchronization

  All OpenText eDirectory servers must maintain accurate time. Time stamps are assigned to each object and property and they ensure the correct order for object and property updates. Using time stamps, OpenText eDirectory determines which replicas need to be synchronized.

- Synchronization tolerances

  Time periods since a server has synced with inbound and outbound data changes, how much data is outstanding, etc.

- Background processes

  Processes that perform a variety of tasks including replication of changes and maintenance of system information.

  - External references
  - Obituaries
  - OpenText eDirectory Schema

Step-by-step instructions for completing these checks are given in the following section, “Checking OpenText eDirectory Health Using OpenText iMonitor” on page 539.

Checking OpenText eDirectory Health Using OpenText iMonitor

Depending on your preference, you can perform an OpenText eDirectory server health check by using either of two methods in OpenText iMonitor:

- Using the Navigator Frame
- Using the Assistant Frame

Using the Navigator Frame

1. Access OpenText iMonitor.

   See “Accessing OpenText iMonitor” on page 221.

2. In the Navigator frame, click the Reports icon .
3. In the Assistant frame, click the **Report Config** link.

   A Runable Report List appears in the Data frame.

4. Click the Configure Report icon  for your desired server information.

   A Server Information Report appears in the Data frame. You will use this report to select the desired options for your report.

5 Check the **Health Sub-Report** check box.

6 To run the report at specified intervals, select the desired options in the Schedule Report section of the Data frame.

IMPORTANT: If you run a scheduled report, it will run as public and might not be able to gather as much information as it would if you ran it as an authenticated user.

7 Click **Run Report** to process the report.

Using the Assistant Frame

1 Access OpenText iMonitor.

See "Accessing OpenText iMonitor" on page 221.

2 In the Assistant frame, click **Agent Health**.

Health check information appears in the Data frame for the server that OpenText iMonitor is reading the information from (not necessarily the server that you are connected to).

Reviewing Report Information

After you have generated a report, the Data frame shows the report results. If you have servers that aren't healthy in your tree, the report is divided into three categories (grouping begins with servers that have the poorest health):

- Servers with warnings
- Servers that are suspect
- Servers that are OK

If none of your servers has warnings or is suspect, those categories are not shown.

For servers that are not healthy, you can click the Agent Health Sub-Report link  next to each server. Use the Online context-sensitive help to resolve the issues. This can help you determine what each of the options means and why it is important, how to resolve any issues, how to adjust the ranges, and whether you want certain options to be included in the health check.

IMPORTANT: If you have a server reported with warnings, we strongly recommend that you resolve the issues with that server. Servers that are suspect should also be evaluated.

For More Information

The tools and techniques used to keep OpenText eDirectory healthy are documented in the OpenText eDirectory Tools & Diagnostics Course 3007. In this course you learn how to

- Perform OpenText eDirectory health checks.
- Perform OpenText eDirectory operations properly.
- Properly diagnose, troubleshoot, and resolve OpenText eDirectory issues.
- Use OpenText eDirectory troubleshooting tools and utilities.

To learn more about this course, visit the OpenText Training Services Web site (https://www.netiq.com/training/).

Resources for Monitoring

The OpenText DSTrace utility runs on Windows and Linux. This tool helps you monitor the vast resources of OpenText eDirectory. For more information on DSTrace, see the following:

- "Configuring Trace Settings" on page 233
- "Looking Into the Directory Services Trace (DSTrace) Options" (http://support.novell.com/techcenter/articles/anp20010801.html)
- "More on Using the DSTrace Command" (http://support.novell.com/techcenter/articles/anp20010901.html)

You can also invest in third-party products that provide additional management solutions for your OpenText eDirectory environment. For more information, see the following Web sites:

- Symantec (http://www.symantec.com/compliance/)
- Blue Lance (http://www.bluelance.com)
- Quest (http://www.quest.com/active-directory/)

If you need to monitor or audit certain characteristics of OpenText eDirectory that our partners do not provide, OpenText Consulting Services can help you use the OpenText Event System for customized assessment and auditing.

Upgrading Hardware or Replacing a Server

This section provides information about transferring or safeguarding OpenText eDirectory on a specific server when you upgrade or replace hardware. It is based on information in "Backing Up and Restoring OpenText eDirectory" on page 417.

The Backup OpenText eDirectory Management Tool allows you to prepare OpenText eDirectory information on a server for

- "Planned Hardware or Storage Device Upgrade without Replacing the Server" on page 541
- "Planned Replacement of a Server" on page 544

Planned Hardware or Storage Device Upgrade without Replacing the Server

If you are planning to upgrade hardware such as a storage device or RAM, you prepare by doing a cold backup of OpenText eDirectory using the Backup eMTool, as well as a file system backup. This will let you safeguard the server's OpenText eDirectory identity and file system data, which has the following benefits:

- If you are replacing storage devices, the backups let you transfer information from the old storage devices to the new.

- If you are replacing the storage device that includes the disk partition/volume containing OpenText eDirectory, having this backup information lets you use the restore process to re-create the OpenText eDirectory database on the new storage device.
- Doing a cold backup of OpenText eDirectory and keeping the database closed afterward means you can upgrade hardware and transfer the database without worrying that the database has changed since the backup.
- If anything goes wrong, you have backups you can use to recover.

For the OpenText eDirectory cold backup, you must use the options to lock and disable OpenText eDirectory on the server, preventing any data change after the backup is made. To other servers that communicate with this server, the server appears to be down. Any OpenText eDirectory information that is normally sent to the server is stored by other servers in the tree until they can communicate with the server again. The stored information is used to synchronize the server when you bring it back online.

NOTE: Because other servers in the OpenText eDirectory tree expect the server to come back online quickly, you should complete the upgrade promptly and open the OpenText eDirectory database on the server as soon as possible.

To perform a planned hardware upgrade:

1. If you are concerned that the upgrade might cause a problem for your server, you might want to prepare another machine to use if necessary.

   See "1. Preparing for a Server Replacement" on page 544.

2. Use a Client command like the following to do a cold backup of the OpenText eDirectory database and keep the database closed and locked when finished. If you use NICI, make sure to back up the security files too.

   ```
   backup -f backup_filename_and_path
   -l log_filename_and_path -t -c -o -d
   ```

   If you use NICI, make sure you back up the NICI files. See "Backing Up Manually with DSBK" on page 442 and "Backup and Restore Command Line Options" on page 446 for more information about using the Client and the switches.

   The OpenText eDirectory database is now locked. You must leave it locked so that no new data changes will be made on that server until you finish the procedure.

   Complete the rest of the procedure promptly, to minimize the amount of time that the server is unavailable.

3. Back up the file system using your backup tool of choice.

   It's important to do this *after* backing up the database, so that the OpenText eDirectory backup files are saved to tape along with the rest of the file system.

4. Down the server and replace the hardware.

5 After replacing the hardware, proceed by following the instructions for the kind of hardware change you made:

If you…	Perform These General Steps
Did not make any changes to storage devices	Bring up the server and unlock the database.
Replaced storage devices, but the disk partition/volume containing OpenText eDirectory was not affected	1. Bring up the server and OpenText eDirectory. 2. Restore the file system only for the disk partitions/volumes that were on the storage devices you changed. 3. Unlock the OpenText eDirectory database.
Replaced the storage device that contained OpenText eDirectory	1. Install the operating system if necessary. 2. Restore the file system on disk partitions that were affected by the storage device change. 3. Install OpenText eDirectory on the new storage device, in a new temporary tree. 4. Restore OpenText eDirectory from backup (which puts it back into the original tree), specifying the option to keep it closed and locked after the restore. Use a command like the following: `restore -r -f` *`backup_filename_and_path`* `-l` *`log_filename_and_path`*. Add the `-u` option if you backed up the files listed in an include file and restore NICI files seperately. 5. Unlock the eDirectory database. 6. If you restored NICI security files, after completing the restore, restart the server to reinitialize the security system. 7. Check to see whether the server responds as usual. Use OpenText iMonitor to check the server and its synchronization. 8. If you were using roll-forward logging on this server, make sure you re-create the roll-forward logs configuration after the restore is complete. After turning on the roll-forward logs, you must also do a new full backup. The settings are reset to the default after a restore, which means roll-forward logging is turned off. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

If the server does not respond as usual, you might need to recover by doing one of the following:

- Re-create the hardware configuration you had before, because it was working before the change.
- Transfer this server's identity to another machine using the file system and OpenText eDirectory backups you made. See “Planned Replacement of a Server” on page 544.

Planned Replacement of a Server

The following instructions are designed for situations where a server is actually replaced by moving the server's OpenText eDirectory identity and file system data onto a different machine. For naming purposes in these instructions, the old server is referred to as Server A, and its replacement is referred to as Server B.

You prepare by doing a cold backup (a backup done while the database is closed) of OpenText eDirectory using the Backup eMTool, as well as a file system backup using your tool of choice. This backup information lets you use the restore process to re-create the server on the new machine.

For the OpenText eDirectory cold backup, you must use the options to lock and disable OpenText eDirectory on Server A, preventing any data change after the backup is made. To other servers that communicate with this server, the server appears to be down. Any OpenText eDirectory information that is normally sent to the server is stored by other servers in the tree until they can communicate with the server again. The stored information is used to synchronize the server when you bring it back online on the new machine, Server B.

NOTE: Because other servers in the OpenText eDirectory tree expect the server to come back online quickly, you should complete the change and restore OpenText eDirectory information on the server as soon as possible.

Follow these general steps to replace a server:

1. To reduce down time for Server A while you are replacing it, it's best to prepare Server B as much as possible before you begin the replacement, by installing the operating system, etc., as described in "1. Preparing for a Server Replacement" on page 544.
2. Do the OpenText eDirectory and file system backups on Server A as described in "2. Creating a Backup of OpenText eDirectory" on page 545.
3. Transfer the information to Server B as described in "3. Restoring OpenText eDirectory Information for a Server Replacement" on page 545.

1. Preparing for a Server Replacement

Use the following checklists for Server A and Server B to determine whether you are ready to replace Server A. Preparing Server B before proceeding will reduce the time the server is down while you transfer from one machine to the other.

Preparation for Server A

- ❒ Make sure that Server A has the latest version of the operating system installed.
- ❒ Make sure the tree for Server A is healthy by running DSRepair on the server that holds the master of the Tree partition and by running time synchronization.
- ❒ Run DSRepair on the database of Server A. Ensure that Server A is synchronized completely.

Preparation for Server B

- ❒ Install the latest version of the operating system. This must be the same operating system as Server A.

❒ Install OpenText eDirectory, putting Server B in a new temporary tree.

(Restoring OpenText eDirectory during "3. Restoring OpenText eDirectory Information for a Server Replacement" on page 545 will put Server B into the original tree that Server A was in.)

Continue with the steps in the next section, "2. Creating a Backup of OpenText eDirectory" on page 545.

2. Creating a Backup of OpenText eDirectory

You must create a backup of OpenText eDirectory prior to a server replacement. After completing "1. Preparing for a Server Replacement" on page 544, use the Client to do a cold backup of the OpenText eDirectory database on Server A, using the advanced options to disable and lock the database after the backup.

To create a cold backup (a backup done while the database is closed) of OpenText eDirectory and keep the database closed afterward:

1. Make sure you have completed "1. Preparing for a Server Replacement" on page 544.
2. Do a cold backup of the OpenText eDirectory database on Server A and keep the database closed and locked when finished, by using a `backup` command like the following in the Client with the `-c`, `-o`, and `-d` switches:

   `backup -f backup_filename_and_path -l log_filename_and_path -t -c -o -d`

   If you use NICI, make sure you back up the NICI files. See "Backing Up Manually with DSBK" on page 442 and "Backup and Restore Command Line Options" on page 446 for more information about using the Client and the switches.

   Server A's OpenText eDirectory database is now locked. You must leave it locked so that no new data changes will be made on that server until you bring it back into the tree by restoring onto Server B.

   Complete the rest of the server upgrade or replacement procedure promptly, to minimize the amount of time that the server is unavailable.
3. Make a full backup of Server A's file system.

   It's important to do the file system backup *after* backing up the database, so that the OpenText eDirectory backup files are saved to tape along with the rest of the file system.

   For complete information on using SMS, see the *Backing Up Data Using SMS* (https://www.novell.com/documentation/open-enterprise-server-2018/bkup_sms_lx/data/hhc3nq5m.html).
4. Lock the OpenText eDirectory database on Server A and unplug Server A from the network.

   Continue with the steps in "3. Restoring OpenText eDirectory Information for a Server Replacement" on page 545.

3. Restoring OpenText eDirectory Information for a Server Replacement

To transfer Server A's OpenText eDirectory identity and file system to Server B:

1. Make sure you have completed "1. Preparing for a Server Replacement" on page 544 and "2. Creating a Backup of OpenText eDirectory" on page 545.
2. Make sure Server B is up and eDirectory is running.

3 Use restore to transfer Server A's OpenText eDirectory identity and file system to Server B:

   3a Copy the OpenText eDirectory cold backup files created for Server A to Server B.

   The backup files can be made much smaller using a third-party file compression tool, because they compress well. This could help you copy the files faster.

   3b Restore the OpenText eDirectory database from Server A onto Server B using the OpenText eDirectory backup files you copied. In the command line client, use a command like the following:

   ```
   restore -r -f backup_filename_and_path -l log_filename_and_path
   ```

   If you use NICI, make sure you restore the NICI files. Add the `-u` option if you backed up files listed in an include file. See "Restoring from Backup Files with DSBK" on page 444 and "Backup and Restore Command Line Options" on page 446 for more information about using the Client and the switches.

   No roll-forward logs need to be included in the restore, because you did a cold backup and kept the database closed afterward. No transactions have occurred in the database because it's closed, so no roll-forward logs have been created since the backup.

   3c Transfer Server A's file system data onto Server B, from backup.

4 If you use NICI, restart the server to reinitialize NICI so it will use the restored NICI security files.

5 Unlock the OpenText eDirectory database.

6 After completing the restore, check to see whether Server B has successfully taken on Server A's identity and is responding as usual. Use OpenText iMonitor to check the server and its synchronization.

   If the server responds as usual, you are finished with the server replacement. You can now uninstall eDirectory from Server A to remove its OpenText eDirectory identity, then use the machine for another purpose. Do not bring Server A back up on the network until you remove OpenText eDirectory, or it will cause confusion in the network with OpenText eDirectory synchronization because Server A and Server B will compete for the same identity.

7 (Conditional) If you were using roll-forward logging on this server, make sure you re-create the roll-forward logs configuration after the restore is complete. After turning on the roll-forward logs, you must also do a new full backup.

   The settings are reset to the default after a restore, which means roll-forward logging is turned off. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

If Server B does not work correctly and you need Server A's identity and file system to be available right away, you can do the following:

1 Unplug Server B's network cable or down the server.

2 Reattach Server A to the network, start it, then open the OpenText eDirectory database.

   Ignore system messages requesting you to run DSRepair.

3 Remove OpenText eDirectory from Server B and try the upgrade again.

Server IP Address Changes

Usually the server's IP address is static. When it changes you need to update the `nds.conf` file for all the OpenText eDirectory instances with the new IP address. `nds.conf` should use the interface name instead of IP address if the IP address changes frequently.

For example: `n4u.server.interfaces=eth0@1524`

After an IP address change, a server's IP-based Key Material Objects (KMO) will not be automatically updated. Though deleting the old KMOs (with IP in their name) is not necessary, it helps to keep the tree clean. Run the `ndsconfig upgrade` command to recreate your KMOs and link them with the NCP Server and LDAP Server objects.

NOTE: Running `ndsconfig upgrade` restarts your OpenText eDirectory instance.

Now the server continues to listen on the new address. Run DSRepair network repair options if there are multiple servers in the tree:

```
ndsrepair -N
```

After running the repair options, restart the OpenText eDirectory server.

For more information on server IP address changes, refer to TID# 3201067

Restoring OpenText eDirectory after a Hardware Failure

A hard disk failure involving the disk partition/volume where OpenText eDirectory is located is equivalent to removing OpenText eDirectory from the server. Fortunately, in a multi-server environment, one server can go down while the rest of the servers in the replica ring remain intact.

To restore OpenText eDirectory after a failure of the disk partition/volume that it resides on, follow the procedures for restoring from your backup files as described in “Preparing for a Restore” on page 436 and “Restoring from Backup Files with DSBK” on page 444.

During the new installation, follow any instructions provided by the manufacturer to verify that the server's hard disks are working. The new hard disk should have at least the same storage capacity as the drive it replaces. Use the local server information files to verify configuration information.

NOTE: ◆We recommend you exclude the DIB directory on your OpenText eDirectory server from any antivirus or backup software processes. Use the OpenText eDirectory Backup Tool to back up your DIB directory. For more information about backing up OpenText eDirectory, see “Backing Up and Restoring OpenText eDirectory” on page 417.

- If you do not have backup files for the server, use the Xbrowse tool to query OpenText eDirectory to help you recover server information. You must do this before you remove the Server object or any associated objects from the tree. Xbrowse and additional information is available from the OpenText Support Web site.

Subtree Search Performance Improvement

The OpenText eDirectory subtree search performance for a large tree with a significantly nested structure remains flat irrespective of the base DN of the search. This has been resolved by using an AncestorID attribute. The AncestorID attribute is a list of entry IDs of all ancestors, associated with each entry. This AncestorID attribute is used internally during the subtree search and therefore restricts the scope of the search.

This attribute gets populated while adding an entry and after upgrade for all the entries in the DIB and is repopulated for all the entries in the subtree after a subtree is moved. However, the subtree search will not use the AncestorID attribute while populating the attribute after upgrade and subtree move. Therefore, the subtree performance remains similar to OpenText pre-eDirectory subtree search performance.

To verify if AncestorIDs are updated after upgrade:

Once the AncestorIDs are populated, the NDS Object Upgrade version changes to 6 or more. You can view this using iMonitor in the **DIB History** section of Agent Information.

To verify if AncestorIDs are updated after the subtree move operation:

While the AncestorIDs are being populated, the attribute UpdateInProgress in the Pseudo Server object has the list of entry IDs of the partition Root of the subtree. Once the AncestorIDs are populated, the attribute will not be present in the Pseudo Server.

DSRepair updates the AncestorID attribute if it is invalid.

Container Readiness

To ensure optimal utilization of the entry cache and enhanced performance of attribute search operations, FLAIM stores attributes with larger values or higher number of values in a separate location namely, Attribute Container. By default the attributes will be moved to the container automatically when the attribute:

- has greater than 25 values
- has a value greater than 2048 bytes

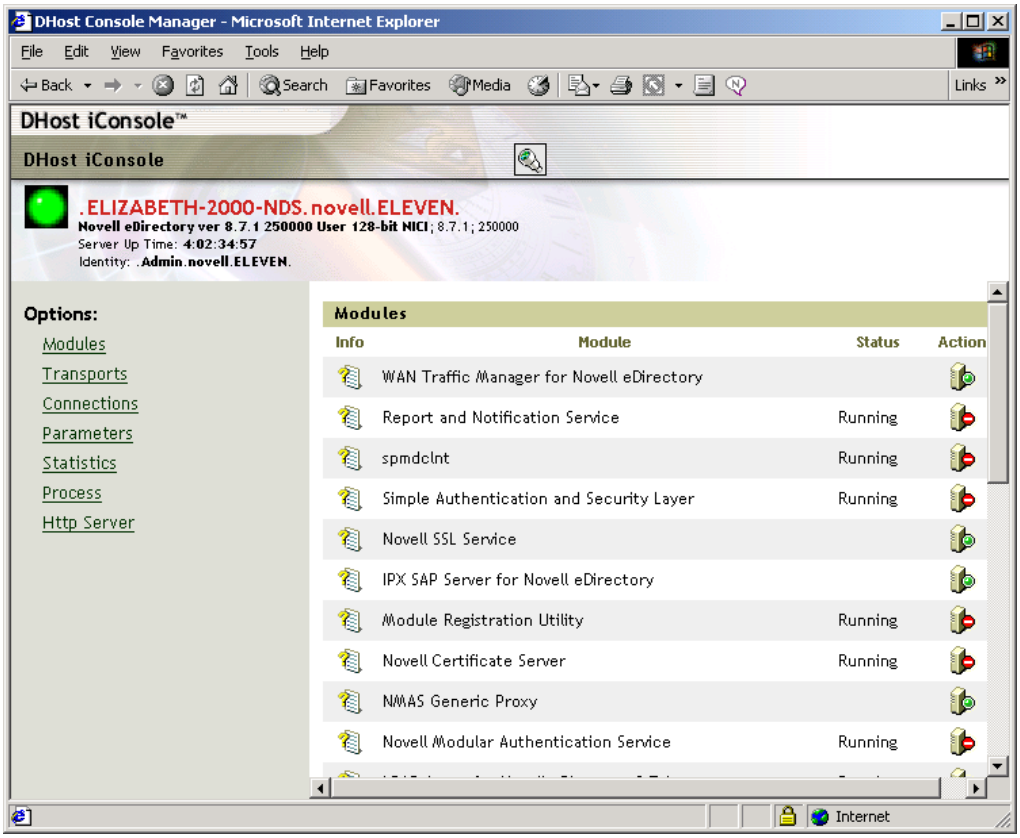
OpenText eDirectory now provides you the flexibility for controlling the movement of attributes to separate attribute containers. An administrator can schedule the attribute movement as required. For more information, see FLAIM Attribute Containerization.

20 DHost OpenText iConsole Manager

DHost OpenText iConsole Manager is a Web-based browser administrative tool that lets you:

- Manage DHost modules
- Query for DHost configuration parameters
- View DHost connection information
- View thread pool statistics
- View details about protocols registered with the DHost protocol stack manager

Figure 20-1 *DHost OpenText iConsole Manager*



This chapter contains the following information:

- "What is DHost?" on page 550
- "Running DHost OpenText iConsole" on page 550
- "Managing OpenText eDirectory Modules" on page 551
- "Querying for DHost Information" on page 553
- "Process Stack" on page 555

What is DHost?

OpenText eDirectory software for Windows and Linux is all built upon the same core code. In order for OpenText eDirectory for Windows and Linux to properly interact with the other versions of OpenText eDirectory, they support a subset of NetWare Core Protocol (NCP) services. This is handled by a program called DHost. DHost sits beneath OpenText eDirectory and provides functionality that the NCP provides naturally.

DHost provides the following services:

Service	Description
NCP Engine	A packet-based protocol that enables a client to send requests to and receive replies from an OpenText eDirectory server. For more information, see the NetWare Core Protocols NDK (https://www.microfocus.com/documentation/open-enterprise-server-developer-documentation/ncp/).
Watchdog	Packets used to make sure workstations are still connected to the OpenText eDirectory server. For more information, see “Watchdog Packet Spoofing”.
Connection Table	A unique number assigned to any process, print server, application, workstation, or other entity that attaches to an OpenText eDirectory server. The number can be different each time an attachment is made. Connection numbers are used in implementing network security and for network accounting. They reflect the objects place in the file servers connection table. Additionally, they provide an easy way to identify and obtain information about the objects logged in on the network.
Event System	Provides a way for applications to monitor the activity of an individual server.
Thread Pool	A sequence of instructions executed as an independent entity and scheduled by system software.
NCP Extensions	Allows server application developers to write NLM™ software to be implemented as NCPs. For more information, see “NCP Extension” (https://www.microfocus.com/documentation/open-enterprise-server-developer-documentation/ncp/ncpdocs/ncpext.htm) in the NCP NDK.
Message Digest	A compressed or condensed form of a document, or an abstract from a document, that functions as a “digital fingerprint” of the larger document. A message digest is used to create a digital signature that is unique to a particular document.

Running DHost OpenText iConsole

- “Running DHost OpenText iConsole on Windows” on page 551
- “Running DHost OpenText iConsole on Linux” on page 551

Running DHost OpenText iConsole on Windows

1. Open a Web browser.
2. In the address (URL) field, enter the following:
   - For http: `http://server.name:port/dhost`
     Example, `http://MyServer:8028/dhost`
   - For https: `https://server.name:port/dhost`
     Example, `https://MyServer:8030/dhost`

   You can also use the server IP address to access the DHost OpenText iConsole. For example:

   `http://137.65.135.150:8028/dhost`
3. Specify a user name, context, and password.

Running DHost OpenText iConsole on Linux

1. Open a Web browser.
2. In the address (URL) field, enter the following:
   - For http: `http://server.name:port/dhost`
     Example, `http://MyServer:8028/dhost`
   - For https: `https://server.name:port/dhost`
     Example, `https://MyServer:8030/dhost`

   You can also use the server IP address to access the DHost OpenText iConsole. For example:

   `http://137.65.135.150:8028/dhost`
3. Specify a user name, context, and password.

Managing OpenText eDirectory Modules

The Modules page in DHost OpenText iConsole provides information about available OpenText eDirectory services and their states. You can also use the Modules page to start and stop (load or unload) these services.

You can only load or unload non-interactive modules such as LDAP, SNMP, and HTTPSTK.

The Modules page has the following attributes:

Attribute	Description
Info	Click  to display the module description, file name, module handle, attributes, and the name of so (shared object) of the selected module.
Module	Displays the module name.
Status	Displays whether the module is running or not.
Action	Indicates whether the module can be run or not. A module can be in one of the following three states: •  indicates that the module is a system module and cannot be unloaded. •  indicates that the module can be loaded and it is ready to load. •  indicates that the module is running.

- “Loading or Unloading Modules on Windows” on page 552
- “Loading or Unloading Modules on Linux” on page 552

Loading or Unloading Modules on Windows

1. Open a Web browser.
2. In the address (URL) field, enter the following:

   `http://server.name:port/dhost`

   for example:

   `http://MyServer:80/dhost`

   You can also use the server IP address to access the DHost OpenText iConsole. For example:

   `http://137.65.135.150:80/dhost`
3. Specify a user name, context, and password.
4. Click **Modules**.
5. Click  to load a module, or  to unload a module.

Loading or Unloading Modules on Linux

1. Open a Web browser.
2. In the address (URL) field, enter the following:

   `http://server.name:port/dhost`

   for example:

```
http://MyServer:80/dhost
```

You can also use the server IP address to access the DHost OpenText iConsole. For example:

```
http://137.65.135.150:80/dhost
```

3 Specify a user name, context, and password.

4 Click **Modules**.

5 Click  to load a module, or  to unload a module.

Querying for DHost Information

Using the DHost OpenText iConsole Manager, you can query for the following information:

- Configuration parameters
- Protocols registered with the PSTACK manager
- Connection properties
- Summary of thread pool

Viewing the Configuration Parameters

Configuration parameters are specific only to Linux.

In the DHost OpenText iConsole Manager, click **Parameters**. See "Running DHost OpenText iConsole on Linux" on page 551 for more information.

The configuration parameters are displayed with the following information:

Option	Description
Parameter name	Displays the name of the configuration parameter.
Default value	Displays the default value of the parameter.
Set value	Displays the value currently set.
Minimum value	Displays the minimum limit that can be set for the parameter.
Maximum value	Displays the maximum limit that can be set for the parameter.
Type	Displays the type of value that can be set for the parameter.

For more information, see "Configuration Parameters" in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

Viewing Protocol Information

In the DHost OpenText iConsole Manager, click **Transports**.

The following protocol information is displayed:

- ID

- Protocol
- Transports

Viewing Connection Properties

In the DHost OpenText iConsole Manager, click **Connections**.

The following connection properties are displayed:

- Conn
- Flags
- Identity
- Display Name
- Transport
- Authentication Name
- SEV Count
- Last Access
- Locked

Viewing the Thread Pools Statistics

In the DHost OpenText iConsole Manager, click **Statistics**.

The following thread pool statistics are displayed:

- Spawned Threads
- Dead Threads
- Idle Threads
- Worker Thread
- Peak Worker Thread
- Ready for Work Thread
- Ready Queue Peak Worker Threads
- Ready Queue Max Wait Time
- Schedule Delay Minimum Time
- Schedule Delay Maximum Time
- Schedule Delay Average Time
- Waiting For Work
- Peaking Waiting For Work

Process Stack

The process stack contains a list of all threads currently running in the DHost process space. You can get detailed information on a thread by clicking the thread ID. This feature is used mainly as a low-level debugging tool for OpenText engineers and support personnel.

This option is available only on Windows.

1. Open a Web browser.
2. In the address (URL) field, enter the following:

   `http://`*`server.name`*`:`*`port`*`/dhost`

   for example:

   `http://MyServer:80/dhost`

   You can also use the server IP address to access the DHost OpenText iConsole. For example:

   `http://137.65.135.150:80/dhost`
3. Specify a user name, context, and password.
4. Click **Process**.
5. To view the call stack for a thread, click the thread ID.

21 Setting the sadmin Password

You can set up a preconfigured admin user that allows access to the HTTP Protocol Stack (HTTPSTK) when OpenText eDirectory is not loaded. The preconfigured admin user, `sadmin`, has rights that are equivalent to the OpenText eDirectory Admin User object. If the server is in a state where OpenText eDirectory is not functioning correctly, you can log in to the server as this user and perform all the diagnostic and debugging tasks necessary that do not require OpenText eDirectory.

NOTE: The sadmin username is case-insensitive.

Use the `ndspassstore` utility to set the sadmin password on Windows and Linux systems.

Enter the following at the server console:

`ndspassstore -a sadmin -w <password>`

where `sadmin` (admin context) is the fully distinguished name of a user having administrative rights and `password` is the password for authentication. Select an appropriate instance in case of a multi-instance scenario.

Example: `ndspassstore -a sadmin -w pass`

ndspassstore is available by default at **C:\Novell\NDS** in Windows and at **/opt/novell/eDirectory/bin** in UNIX.

22 The OpenText eDirectory Management Toolbox

The OpenText eDirectory Management Toolbox (eMBox) lets you access all of the OpenText eDirectory back-end utilities remotely, as well as on the server.

The eMBox works with OpenText Identity Console to provide Web-based access to OpenText eDirectory utilities such as DSRepair, DSMerge, Backup and Restore, and Service Manager.

RAC must be configured for the following tasks that are under OpenText eDirectory maintenance in the OpenText Identity Console:

- Backup Configuration
- Graft Tree
- Repair OpenText eDirectory
- Repair Server
- Repair Sync
- Replica Repair
- Replica Ring Repair
- Restore
- Schema Maintenance
- Service Manager
- Merge Tree
- Rename Tree

All functions are accessible, either on the local server or remotely, through a command line client. You can perform tasks for multiple servers from one server or workstation using the Client.

For all OpenText eDirectory Management Tools (eMTools), to run, including Backup, DSRepair, DSMerge, Schema Operations, and OpenText eDirectory Service Manager, eMBox must be loaded and running on the OpenText eDirectory server.

NOTE: For more information on using the eMTools, see “Troubleshooting Utilities on Linux” on page 834.

In this section:

- “Using the Command Line Client” on page 560
- “Using the Logger” on page 569
- “Using the eMBox Client for Backup and Restore” on page 570

Using the Command Line Client

One way to access is to use its Java command line client. The command line client has two modes: interactive and batch. In the interactive mode, you run the commands one at a time. In the batch mode, you can run a group of commands unattended. The command line client has logging service for both modes.

The command line client is a Java application. To run it, you must install the latest version of Azul ZuluOpenJDK (1.8 or above). You must also ensure to upgrade any older version of Java by installing the patch upgrades available. Once you have the latest version of Java installed, export any of the following environment variables:

- `EDIR_JAVA_HOME`
- `JAVA_HOME`
- `JRE_HOME`

NOTE: ◆On Linux, if none of the above mentioned environment variables are found, command line client searches for the Java binary in the default PATH environment variable.

- OpenText eDirectory 9.1 SP2 and above supports Azul ZuluOpenJDK 1.8.0_192.

Examples

Few examples for the environment variables are mentioned below:

- **Linux**
  - `EDIR_JAVA_HOME=/usr/java/java1.8.0_131`
  - `JAVA_HOME= /usr/java/java1.8.0_131`
  - `JRE_HOME= /usr/java/java1.8.0_131/jre`
- **Windows**
  - `EDIR_JAVA_HOME= C:\Program Files\Java\jdk1.8.0_131`
  - `JAVA_HOME= C:\Program Files\Java\jdk1.8.0_131`
  - `JRE_HOME= C:\Program Files\Java\jdk1.8.0_131\jre`

You must also have access behind the firewall to the servers you want to manage. You can perform tasks for multiple servers from one server or workstation.

NOTE: The OpenText eDirectory Management Toolbox only supports English, both in the command line client and command line help.

In this section:

- “Displaying the Command Line Help” on page 561
- “Running the Command Line Client in Interactive Mode” on page 561
- “Running the Command Line Client in Batch Mode” on page 565
- “eMBox Command Line Client Options” on page 567

- "Establishing a Secure Connection with the Client" on page 568
- "Finding Out OpenText eDirectory Port Numbers" on page 568

Displaying the Command Line Help

To display the general command line help before going in to the Client, do the following:

- Linux: At the command line, enter `edirutil -?`.
- Windows: Run *`drive`*`\novell\nds\\edirutil.exe -?`

To display the interactive command line help while you are in the interactive mode, at the Client prompt enter a question mark (?). For example, `Client> ?`

The help displays information on the command line options like the information in "eMBox Command Line Client Options" on page 567.

Running the Command Line Client in Interactive Mode

Interactive mode lets you run commands one at a time.

In this section:

- "Running the Client on an OpenText eDirectory Server" on page 561
- "Running the Client on a Workstation" on page 562
- "Setting Up the Path and Classpath for Client" on page 562
- "Logging In to a Server" on page 563
- "Setting Preferred Languages, Timeout, and Log File" on page 563
- "Listing eMTools and Their Services" on page 563
- "Running a Particular Service" on page 564
- "Logging Out From the Current Server" on page 564
- "Exiting the Client" on page 565

Running the Client on an OpenText eDirectory Server

The Client and Sun JVM 1.3.1 are installed with OpenText eDirectory. To open the Client in interactive mode on an OpenText eDirectory server, do the following:

- Linux: At the command line, enter `edirutil -i`.
- Windows: Run *`drive`*`\novell\nds\edirutil.exe -i`

The `edirutil` file gives you a shortcut to running the Client. It points to the Java executable and the default location where the Client is installed with OpenText eDirectory. You can also enter the information manually, as described in "Setting Up the Path and Classpath for Client" on page 562.

You must have access behind the firewall to use the command line client for the servers you want to manage—so if you are remote, you'll need VPN access.

Running the Client on a Workstation

To use the Client on a machine other than an OpenText eDirectory server:

- Copy the `eMBoxClient.jar` file from an OpenText eDirectory server to your machine.
  - Windows: `\novell\nds\\eMBoxClient.jar`
  - Linux: `/opt/novell/eDirectory/lib/nds-modules//eMBoxClient.jar`
- Make sure the machine has Sun JVM 1.3.1 installed.
- Make sure you have access behind the firewall to use the command line client for the servers you want to manage.

You can't use the `edirutil` command on a workstation as a shortcut to getting in to the Client in interactive mode as you can on a server. You must either set up the environment once in your path and class path, or enter it manually each time. See "Setting Up the Path and Classpath for Client" on page 562.

Setting Up the Path and Classpath for Client

If you are running the Client on an OpenText eDirectory server and have not changed the location of Java or the `eMBoxClient.jar` file, you can use `edirutil` as a shortcut to running the Client. See "Running the Client on an OpenText eDirectory Server" on page 561.

But if you have changed the default locations, or you are running the `eMBoxClient.jar` file on a machine that is not a server, or you want to enter the classpath manually, you need to set up the path and classpath for the Client as explained in this section.

You can run the Client from anywhere on your machine if you do the following:

- Add to your path the directory where the Java executable (for example, `java.exe`) is located, or make sure that Java is already running.

  If you are on a server, this is probably already done for you. On Windows, Linux, and UNIX servers, the directory needs to be in your path.

  On a workstation, you might need to set it up yourself. For example, in Windows, click **Start** > **Settings** > **Control Panel** > **System**. On the **Advanced** tab, click **Environment Variables** and add the path to the **Path** variable.

  To enter this manually: If the path to the Java executable has not been added to your path, at the command line you will need to first change to the directory containing the Java executable before running. For example, in Windows enter `cd c:\novell\nds\\jre\bin`
- Add the path to the `eMBoxClient.jar` file to your classpath.

  Windows server or workstation: `set CLASSPATH=`*`path`*`\eMBoxClient.jar`

  Linux server or workstation: `export CLASSPATH=`*`path/`*`eMBoxClient.jar`

  To enter this manually: An alternative way to specify the classpath is to use the `-cp` flag for Java each time you want to run:

  `java -cp `*`path/`*`eMBoxClient.jar -i`

  For example, in Windows enter `java -cp c:\novell\nds\\eMBoxClient.jar -i`

After doing both of these steps, you can run the client in interactive mode from anywhere on your machine using the following command:

```
java  -i
```

For information on Java commands, see the Java documentation on the Oracle Web site (http://www.oracle.com/technetwork/java/).

Logging In to a Server

To log in to a server, you need to specify the server name or IP address and the port number to connect to a particular server. A user name and password are not needed for public logins.

For example, after opening the Client in interactive mode, enter

```
login -s 137.65.123.244 -p 8030 -u admin.mycompany
-w mypassword
```

Use `-n` to make non-secure connection with the client as shown in the below example.

```
login -s 137.65.123.244 -p 8028 -u admin.mycompany
-w mypassword -n
```

For more information about port numbers, see "Finding Out OpenText eDirectory Port Numbers" on page 568.

Setting Preferred Languages, Timeout, and Log File

The default language is the client system language, so in most cases you won't need to explicitly set a language. Similarly, the default timeout should work in most cases. To set the log file, specify the filename and the mode for opening it (append or overwrite).

See the following table for sample commands.

Command	Description
`set -L en,de`	Sets the language preference to English and German (in that order).
`set -T 100`	Sets the timeout to 100 seconds. The timeout setting specifies how long to wait for responses from the server.
`set -l mylog.txt -o`	Uses `mylog.txt` as the log file and overwrites when opening it. Default=append

Listing eMTools and Their Services

After logging in to a server, you can use the list command to display a list of the services available on that server.

The list command displays the following eMTools and their services dynamically:

eMTool	Description
Backup	OpenText eDirectory Backup eMTool
DSMerge	OpenText eDirectory Merge eMTool
DSRepair	OpenText eDirectory Repair eMTool
DSSchema	OpenText eDirectory Schema Operations eMTool
service	OpenText eDirectory Service Manager eMTool

Use `-r` to force the refresh of the list. Use `-t` to list service details. Use `-f` to list just the command format.

See the following table for sample commands.

Command	Description
`list`	Lists the eMTools available on the server.
`list -r`	Refreshes the eMTool list.
`list -t backup`	Lists Backup services with details.
`list -t dsrepair`	Lists DSRepair services with details.
`list -t dsmerge -f`	Lists DSMerge services with command formats only.

Running a Particular Service

You can perform tasks using each of the eMTool services after you have logged in to a server. For example:

Command	Description
`dsrepair.rld`	Repair local database.
`backup.getconfig`	Get backup configuration information.

For more information, see the following:

- “Using the eMBox Client for Backup and Restore” on page 570
- “Using the Client to Merge Trees” on page 291
- “Using the Client to Repair a Database” on page 333
- “Using the Client Service Manager eMTool” on page 190

Logging Out From the Current Server

To log out from the current session, use the following command:

```
logout
```

If you log in to a different server, you don't need to use this command. You are automatically logged out of the current server.

Exiting the Client

To exit the client, use either of the following commands:

```
exit
```

or

```
quit
```

Running the Command Line Client in Batch Mode

There are three ways you can run the Client in batch mode:

- “Single Tasks” on page 565
- “Internal Batch File” on page 565
- “System Batch File” on page 566

You can use a combination of the system and internal batch files for more flexibility and for organizing and reusing commands that you run often.

Single Tasks

You can perform a single task in batch mode at the command line, simply by entering the command using the `-t` option to specify the tool and task, and omitting the `-i` option (`-i` specifies interactive mode). For example,

```
java -s 137.65.123.244 -p 8008 -u admin.mycompany
-w mypassword -l mylog.txt -t dsrepair.rld -n
```

For multiple tasks on different servers, or for tasks you perform often, a better alternative is to use an internal batch file. For more information, see the following section, “Internal Batch File” on page 565.

Internal Batch File

To run the Client in batch mode using a Client internal batch file, you need to create a file which contains a group of commands you would run in the interactive mode.

A Client internal batch file lets you run all the commands in the batch file without your attention. You can perform multiple tasks with multiple tools on the same server without logging in and logging out again for each task. From one server, you can also perform tasks with multiple tools on multiple servers.

Internal batch files can help you organize and reuse commands that you perform often, so you don't need to enter them manually at the command line each time.

You can go to the command line and run the internal batch file using a Client command. For example, this command logs in to a server and runs the commands listed in the `mybatch.mbx` file:

```
java -s 137.65.123.244 -p 8008 -u admin.mycompany -w mypassword -l
mylog.txt -o -b mybatch.mbx -n
```

Another option is to put the same kind of command in a system batch file, so that you can schedule it to run on the server unattended. See “System Batch File” on page 566.

Here is an example of an internal batch file. It contains examples of the commands you could run and an example of logging in to a different server. This example assumes that you logged in to a server when you opened the Client. Each command must be on a separate line. Lines beginning with # are comments.

```
# This file is named mybatch.mbx.
# This is an example of commands you could use in
# an internal command batch file.

# Backup commands
backup.getconfig
backup.backup -b -f mybackup.bak -l backup.log -t -w

# DSRepair commands
dsrepair.rld

# Log in to a different server
login -s 137.65.123.255 -p 8008 -u admin.mycompany -w mypassword -n

# DSMerge commands
dsmerge.pr -u admin.mycompany -p admin.mycompany -n mypassword # Schema
Operations
dsschema.rst
dsschema.dse
dsschema.rls
dsschema.gsu
dsschema.scc
dsschema.irs -n LocalTree

# DSService commands
service.serviceList

# End of example.
```

System Batch File

As with other command line tools, you can create system batch files containing Client commands and run them manually at the command line or schedule them to run on the server unattended. For example, you can run backups unattended, using system batch files like the examples described in “Doing Unattended Backups, Using a Batch File with the eMBox Client” on page 573.

From one server, you can perform tasks with multiple tools on multiple servers.

In a system batch file, you can use a combination of Client single commands and internal batch files for more flexibility and for organizing and reusing commands that you run often. For more information, see “Internal Batch File” on page 565 above.

Consult the documentation for your operating system or third-party scheduling software for instructions on how to run batch files unattended.

eMBox Command Line Client Options

Option	Description
`-?` or `-h`	Display help information
`-i`	Interactively run commands one at a time.
`-s server`	Name or IP address of the server. Default=`127.0.0.1`
`-p port`	Port number of the server. Default=`8008`
`-u user`	User DN. For example, `admin.mycompany`. Default=`anonymous`
`-w password`	Password associated with the user specified with `-u`.
`-m mode`	Login mode. Default=`dclient`
`-n`	Do not try to make a secure SSL connection. Use a nonsecure connection. If you do not use this option, the Client will try to establish an SSL connection, and you must have the JSSE files in your class path or it will return an error. See "Establishing a Secure Connection with the Client" on page 568 for more information.
`-l log file`	Name of the log file.
`-o`	Overwrite the log file when opening it.
`-T timeout`	How long (in seconds) to wait for responses from the server.
`-L language`	List of comma-delimited acceptable languages in order of preference, such as `en-US,de_DE`. This option defaults to the client system language.
`-t [tool.]task options`	Perform a single service with this connection. The string following `-t` should be a valid command.
`-b batch file`	Perform a group of services as specified in the batch file. The commands in the batch file should be put on separate lines. Lines preceded by # are comments.

Establishing a Secure Connection with the Client

If you use a nonsecure connection, all the information you enter, such as user names and passwords, is sent over the wire in clear text.

If you instead want to establish a secure connection using SSL, do the following:

- Make sure you don't use the `-n` option in your command when logging in to a server. It specifies a nonsecure connection. A secure connection is the default.
- Make sure you have the following Java Secure Socket Extension (JSSE) files in your class path:
  - `jsse.jar`
  - `jnet.jar`
  - `jcert.jar`

  If you don't, the Client will return an error saying that it cannot establish a secure connection.

  You can get these files and information about JSSE from the Oracle Web site (http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html).

Finding Out OpenText eDirectory Port Numbers

When logging in to a server in the Client, you must specify a port number.

If you specified a port number when you installed OpenText eDirectory, use that number.

For all platforms, the default nonsecure port is 8008, and the default secure port is 8030.

The following sections give some additional tips for finding out the port that is assigned to OpenText eDirectory:

- "On Windows" on page 568
- "On Linux" on page 568

On Windows

1. Click **Start** > **Settings** > **Control Panel**.
2. Double-click the **OpenText eDirectory Services** icon, then click the **Transport** tab.
3. Look up the secure or nonsecure port.
   - For the nonsecure port, click the plus sign next to HTTP.
   - For the secure port, click the plus sign next to HTTPS.

   Click the plus sign next to **Bound Transports** to see the port number.

On Linux

You can use this command to see a list of ports:

```
ndsconfig get | grep http
```

Look for the lines that say http.server.interface and then a port number.

Using the Logger

The Logger is an infrastructure module that logs all the events for all the OpenText eDirectory Management Tools (eMTools) such as DSBackup, DSMerge, and DSRepair. In this release, only one log file is provided in which all eMTools log their operations.

The Logger is different than the client logging service, which is provided through the log files that you specify when you run the client. For example, when you specify `-l mylogfile.txt` in a Client command or when you enter `mylogfile.txt` as a log file name in OpenText Identity Console. The Logger currently records all server messages for tasks that are performed by the eMTools, showing greater detail. By contrast, the client logging service records client messages and messages sent to the client, which give a general report of progress.

Logging is asynchronous, and all operations are logged by default.

This release of the Logger provides the following features:

- The ability to change the log file name and location.

  By default, log files are created in the `\log` directory located in the same directory that OpenText eDirectory was installed in.
- The ability to change the maximum file size, after which the log file will reset.

  The maximum file size is 8 MB.
- The ability to change the logging mode.

  You can choose to append all new messages to the log file or to overwrite an existing log file. The Append option is set by default.
- The ability to start and stop the logging.

  By default, the logger is in Start mode when the starts up. While in Stop mode, no messages are logged.
- The ability to reset the log file contents.
- The ability to read the log file from a client machine.

For more information see:

- “Using the Logger Command Line Client” on page 569

Using the Logger Command Line Client

The following table lists the Logger command line client options:

Option	Description
`logstart`	Starts the logger.
`logstop`	Stops the logger.
`readlog`	Displays the current log file.
`getlogstate`	Displays the current state of the logger (Start/Stop).

Option	Description
`getloginfo`	Displays the name, logging mode (Append/Overwrite), maximum size, and the current size of the log file.

`setloginfo [-f filename] [-s size in Kilo bytes] [-a | -o]`	Lets you set the name, size, and logging mode (Append/Overwrite) of the log file using the following parameters:
	`-f filename`
	The log file name.
	`-s size in KB`
	The maximum size of the log file.
	◆ `-a`
	New log messages will be appended to the current one.
	◆ `-o`
	The log file will be overwritten.

`emptylog`	Clears the contents of the server log file.

Using the eMBox Client for Backup and Restore

The eMBox Client is a command line Java client that gives you access to eMBox tools such as the OpenText eDirectory Backup eMTool. You can back up, restore, and configure roll-forward logging for multiple servers from a single machine if you have access behind the firewall. You can perform advanced tasks remotely using the eMBox Client, a command line Java client, with access behind the firewall or through a VPN.

The OpenText eDirectory Backup Tool is part of the eMBox tool set. The eMBox is a service that is installed on the server as part of OpenText eDirectory.

The Backup Tool comprises the following files:

Filename	Description
`backupcr`	Core library that contains all backup and restore functionality. This library has no user interface. It is loaded and linked dynamically by the backuptl program.
`backuptl`	Tool interface to the backupcr library. Provides backup and restore functionality through the DSBK architecture. This can be accessed via the OpenText Identity Console or DSBK, the Java command line client.
`dsbackup_en.xlf`	Language file containing messages returned by the Backup Tool.

IMPORTANT: The restore verification process is backward compatible only with OpenText eDirectory 8.5 or later. If you want to use the new backup and restore on servers that participate in a replica ring, make sure you upgrade all the servers in the replica ring to OpenText eDirectory 8.5 or later.

Because the eMBox Client can be run in batch mode, you can use it to do unattended backups using the OpenText eDirectory Backup eMTool.

The `eMBoxClient.jar` file is installed on your server as part of OpenText eDirectory. You can also copy the file and run it on any machine with Sun JVM 1.3.1. For more information, see "The OpenText eDirectory Management Toolbox" on page 559 and "Running the Client on a Workstation" on page 562.

Before performing backup and restore tasks, review "Checklist for Backing Up OpenText eDirectory" on page 418 for an overview of the issues involved in planning an effective OpenText eDirectory backup strategy.

- "Prerequisites" on page 571
- "Backing Up Manually with the eMBox Client" on page 572
- "Doing Unattended Backups, Using a Batch File with the eMBox Client" on page 573
- "Configuring Roll-Forward Logs with the eMBox Client" on page 574
- "Restoring from Backup Files with the eMBox Client" on page 576

Prerequisites

- ❒ Make sure the `eMBoxClient.jar` file is on the machine you want to initiate the backup from.

  The file is installed on your server as part of OpenText eDirectory installation. You can copy it from there and run it on any machine with Sun JVM 1.1.3. You can run backups for multiple servers from a single machine if you have access behind the firewall. For more information, see "Using the Command Line Client" on page 560.

- ❒ If you are planning to use roll-forward logs for this server, make sure they are turned on before a backup is made.

  You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open.

  For more information on roll-forward logs, see "Using Roll-Forward Logs" on page 432. For how to turn them on, see "Configuring Roll-Forward Logs with the eMBox Client" on page 574.

- ❒ Review the description of the command line options in "Backup and Restore Command Line Options" on page 446.

- ❒ For multiple-server trees, you should upgrade all the servers that share replicas with this server to OpenText eDirectory 8.5 or later.

Backing Up Manually with the eMBox Client

Use the eMBox Client to back up data from an OpenText eDirectory database to a file you specify on the server where the backup is being performed. This backup file or set of files contains information necessary to restore OpenText eDirectory to the state it was in at the time of the backup. The results of the backup process are written to the log file you specify.

To back up the OpenText eDirectory database on a server using the eMBox Client:

1 Run the eMBox Client in interactive mode.

- Linux: At the command line, enter `edirutil -i`.
- Windows: Run *`drive`*`\novell\nds\edirutil.exe -i`

The `edirutil` file gives you a shortcut to running the eMBox Client. It points to the Java executable and the default location where the eMBox Client is installed with eDirectory.You can also enter the information manually, as described in "Setting Up the Path and Classpath for Client" on page 562.

When the eMBox Client opens, the eMBox Client prompt appears: eMBox Client>

2 Log in to the server you want to back up by entering

`login -s` *`server_name_or_IP_address`* `-p` *`port_number`* `-u` *`username.context`* `-w` *`password`*

For example, in Windows enter

```
login -s 151.155.111.1 -p 8009 -u admin.mycompany -w mypassword
```

If you get an error saying that a secure connection cannot be established, make sure your machine has the JSSE files listed in "Establishing a Secure Connection with the Client" on page 568.

For help finding out which port number to use, see "Finding Out OpenText eDirectory Port Numbers" on page 568.

The eMBox Client indicates whether the login is successful.

3 Enter the backup command at the eMBox Client prompt, following this general pattern:

`backup -b -f` *`backup_filename_and_path`* `-l` *`backup_log_filename_and_path`* `-u` *`include_file_filename_and_path`* `-t -w -a`

A space must be between each switch. The order of the switches is not important.

For example, in Windows enter

```
backup -b -f c:\backups\8_20_2001.bak -l c:\backups\backup.log -u
c:\backups\myincludefile.txt -t -w -a
```

This example command would result in a full backup (`-b`) with the backup file placed at `c:\backups\8_20_2001.bak` and the log file for the process placed at `c:\backups\backup.log`. This command specifies that other files should be backed up along with the database:

- The files listed in an include file (`-u c:\backups\myincludefile.txt`) that was created beforehand by the administrator.
- Stream files (`-t`)

This example command specifies that the backup file should be overwritten (`-w`), so if a file of the same name existed, the Backup eMTool would replace it.

- `-a` option removes the old log files from the roll-forward log directory during a hot continuous back-up.

The eMBox Client indicates whether the backup is successful.

4 Log out from the server by entering the following command:

```
logout
```

5 Exit the eMBox Client by entering the following command:

```
exit
```

6 Make sure you do a file system backup shortly after the OpenText eDirectory backup is created, to put the OpenText eDirectory backup files safely on tape. The Backup eMTool only places them on the server.

For more information on manual backup, refer to "Backing Up Manually with DSBK" on page 442.

Doing Unattended Backups, Using a Batch File with the eMBox Client

Use a batch file to do unattended backups of OpenText eDirectory through the eMBox Client. For example, you might want to do a full backup of OpenText eDirectory on your servers weekly and an incremental backup nightly.

You can run the eMBox Client in batch mode using a system batch file, an eMBox Client internal batch file, or a combination of both. For more information, see "Running the Command Line Client in Batch Mode" on page 565.

This procedure describes using a system batch file:

1 Create a system batch file to back up the servers, following these general patterns, with one line per server.

In Windows and Linux environments, this is the general pattern:

```
java -cp path/eMBoxClient.jar embox -s server_name -p port_number -u
username.context -w password -t backup.backup -b -f
backup_filename_and_path -l backup_log_filename_and_path -u
include_file_filename_and_path -t -w
```

For examples and more explanation, see "Example of System Batch Files for Unattended Backups" on page 574.

For nightly incremental backups, you could use the same file you use for full backups, but change the `-b` switch to `-i` to do an incremental backup instead of a full backup. It's also probably a good idea to use a different backup filename for incremental backups than for the full backup.

For help finding out which port number to use, see "Finding Out OpenText eDirectory Port Numbers" on page 568. If you want to use a secure connection, see "Establishing a Secure Connection with the Client" on page 568. For information on using an eMBox Client internal batch file as well, see "Running the Command Line Client in Batch Mode" on page 565.

2 Run the batch files unattended, according to the instructions in your operating system or third-party documentation.

3 Make sure you schedule file system backups shortly after OpenText eDirectory backups, to place OpenText the eDirectory backup files safely on tape.

   The Backup eMTool only places them on the server.

4 Periodically check the results recorded in the log file you specified, to make sure the unattended backups are successful.

Example of System Batch Files for Unattended Backups

Below is an example system batch file:

Example Batch File for Windows

```
java -cp c:\novell\nds\embox\eMBoxClient.jar embox -s myserver -p 8008 -u
admin.myorg -w mypassword -n -t backup.backup -b -f c:\backup\backup.bak -
u c:\backup\includes\includefile.txt -l c:\backup\backup.log -t -w
```

In this example batch file, the following options are shown.

- A full backup is specified (`-b`).
- An include file is specified (`-u`). This is optional. You can use an include file if you want to back up other files of your choice. The include file must be created beforehand.
- Stream files (`-t`) are also backed up.
- The option to overwrite a backup file of the same name is specified (`-w`).

  IMPORTANT: If a backup file of the same name exists (this is likely if you use the same batch file regularly), it's important to use the `-w` option to overwrite the existing backup file to make sure your backup is successful.

  In batch mode, if `-w` is not specified and a file of the same name exists, the default behavior is to not overwrite the file, so a backup will not be created. In interactive mode, if `-w` is not specified, the eMBox Client will ask you whether you want to overwrite the file.

  If you are making a file system backup shortly after each full or incremental backup of OpenText eDirectory, your previous backup files should have been copied from the server to file system backup tapes, so it should be safe to use this option to overwrite the existing backup file.

- A nonsecure port is used in this example (`-p 8008`), so a nonsecure connection is specified (`-n`).

Configuring Roll-Forward Logs with the eMBox Client

Use the eMBox Client to change the settings for roll-forward logs. You can do the following tasks:

- Find out the current settings
- Turn roll-forward logging on or off

  You must turn on roll-forward logging for servers that participate in a replica ring. If you don't, when you try to restore from your backup files you will get errors and the database will not open.

- Change the roll-forward logs directory
- Set the minimum and maximum roll-forward log size

- Find out the current and last unused roll-forward log
- Turn stream file logging on or off for the roll-forward logs

For information about roll-forward logging, see "Using Roll-Forward Logs" on page 432.

1. Run the eMBox Client in interactive mode:
   - Linux: At the command line, enter `edirutil -i`.
   - Windows: Run *drive*`\novell\nds\edirutil.exe -i`.

   The `edirutil` file gives you a shortcut to running the eMBox Client. It points to the Java executable and the default location where the eMBox Client is installed with OpenText eDirectory. it includes the necessary `-ns` option. You can also enter the options manually, as described in "Running the Client on a Workstation" on page 562.

   When the eMBox Client opens, the eMBox Client prompt appears: eMBox Client>
2. Log in to the server you want to configure roll-forward logging on by entering

   `login -s` *`server_name_or_IP_address`* `-p` *`port_number`* `-u` *`username.context`* `-w` *`password`*

   For example, in Windows, enter

   ```
   login -s 151.155.111.1 -p 8009 -u admin.mycompany -w mypassword
   ```

   If you get an error saying that a secure connection cannot be established, make sure your machine has the JSSE files listed in "Establishing a Secure Connection with the Client" on page 568.

   For help finding out which port number to use, see "Finding Out OpenText eDirectory Port Numbers" on page 568.

   The eMBox Client indicates whether the login is successful.
3. (Optional) Find out the current settings by entering the following command:

   `getconfig`

   No switches are necessary.

   The following is an example of the information you receive:

   ```
   Roll forward log status OFF
   Stream file logging status OFF
   Current roll forward log directory C:\rfl\nds.rfl
   Minimum roll forward log size (bytes) 104857600
   Maximum roll forward log size (bytes) 4294705152
   Last roll forward log not used 00000000.log
   Current roll forward log 00000001.log
   *** END ***
   ```
4. Change the settings using the setconfig command, following this general pattern:

   `setconfig [-L|-l] [-T|-t] -r` *`path_to_roll-forward_logs`* `-n` *`minimum_file_size`* `-m` *`maximum_file_size`*

   A space must be between each switch. The order of the switches is not important.

   Ideally, you would have a separate disk partition/volume dedicated to roll-forward logs to make it easier to monitor disk space and rights.

WARNING: If you turn on roll-forward logging, don't use the default location. For fault tolerance, put the directory on a different disk partition/volume and storage device than OpenText eDirectory. The roll-forward logs directory must be on the server where the backup configuration is being changed.

IMPORTANT: If you turn on roll-forward logging, you must monitor disk space on the volume where you place the roll-forward logs. If left unchecked, the log file directory will grow until it fills up the disk partition/volume. If roll-forward logs cannot be created because no more disk space is available, OpenText eDirectory stops responding on that server. We recommend you periodically back up and remove unused roll-forward logs from your server. See "Backing Up and Removing Roll-Forward Logs" on page 435.

5 Log out from the server by entering the following command:

```
logout
```

6 Exit the eMBox Client by entering the following command:

```
exit
```

Restoring from Backup Files with the eMBox Client

Use the eMBox Client to restore an OpenText eDirectory database from data stored in backup files you created manually or with a batch file. The results of the restore process are written to the log file you specify.

The eMBox Client also lets you use advanced restore options not available in OpenText Identity Console. They are described in "Backup and Restore Command Line Options" on page 446, under restore and restadv.

To restore an OpenText eDirectory database on a server using the eMBox Client:

1 Make sure you have gathered the backup files you need, as described in "Preparing for a Restore" on page 436.

2 Run the eMBox Client in interactive mode:

- Linux: At the command line, enter `edirutil -i`.
- Windows: Run *drive*`\novell\nds\edirutil.exe -i`

The `edirutil` file gives you a shortcut to running the eMBox Client. It points to the Java executable and the default location where the eMBox Client is installed with OpenText eDirectory, it includes the necessary -ns option. You can also enter the information manually, as described in "Running the Client on a Workstation" on page 562.

When the eMBox Client opens, the eMBox Client prompt appears: eMBox Client>

3 Log in to the server you want to restore by entering

```
login -s server_name_or_IP_address -p port_number -u username.context -w password
```

For example, in Windows enter

```
login -s 151.155.111.1 -p 8009 -u admin.mycompany -w mypassword
```

If you get an error saying that a secure connection cannot be established, make sure your machine has the JSSE files listed in "Establishing a Secure Connection with the Client" on page 568.

For help finding out which port number to use, see "Finding Out OpenText eDirectory Port Numbers" on page 568.

The eMBox Client indicates whether the login is successful.

4 Enter the `restore` command at the eMBox Client prompt, following this general pattern:

`restore -r -a -o -f` *`full_backup_path_and_filename`* `-d` *`roll-forward_log_location`* `-l` *`restore_log_path_and_filename`*

A space must be between each switch. The order of the switches is not important. Make sure you use the `-r` switch to restore the OpenText eDirectory database itself. Otherwise only the other kinds of files will be restored. If you want the database to be active and open when the restore is complete, make sure you specify `-a` and `-o`.

If you are restoring roll-forward logs, make sure you include the full path to the logs, including the directory that is automatically created by OpenText eDirectory, usually named `\nds.rfl`. For more information about this directory, see "Location of the Roll-Forward Logs" on page 434.

For example:

```
restore -r -a -o -f sys:/backup/nds.bak -d $HOME/rfldir/nds.rfl -l
$HOME/backups/backup.log
```

This example command specifies that the database itself should be restored (`-r`), and it should be activated (`-a`) and opened (`-o`) after the restore verification is successfully completed. The `-f` switch indicates where the full backup file is, `-d` the roll-forward logs, and `-l` the log file in which to record the results of the restore.

The eMBox Client will restore the full backup, then prompt you for the incremental backup files.

5 (Conditional) If you are restoring incremental backup files, provide the path and filename for each one when the eMBox Client prompts you for the next incremental file.

It will tell you the ID of the next file, which you can find in the incremental backup file header.

The eMBox Client indicates whether the restore was successful.

6 (Conditional) If the restore was not successful, check the log file to see the errors.

If the restore verification fails, see "Recovering the Database If Restore Verification Fails" on page 456.

NOTE: If the server you are restoring shares a replica with a server running an earlier version than OpenText eDirectory 8.5, the restore log will show a -666 error (incompatible DS version) for that replica.

---

7 Log out from the server by entering the following command:

```
logout
```

8 Exit the eMBox Client by entering the following command:

```
exit
```

9 (Conditional) If you restored NICI security files, after completing the restore, restart the server to reinitialize NICI and then restore DIB.

10 Make sure the server is responding as usual.

11 (Conditional) If you are using roll-forward logging on this server, you must re-create your configuration for roll-forward logging to make sure it is turned on and the logs are being saved in a fault-tolerant location. After turning on the roll-forward logs, you must also do a new full backup.

This step is necessary because during a restore, the configuration for roll-forward logging is set back to the default, which means that roll-forward logging is turned off and the location is set back to the default. The new full backup is necessary so that you are prepared for any failures that might occur before the next unattended full backup is scheduled to take place.

For more information about roll-forward logs and their location, see “Using Roll-Forward Logs” on page 432.

Your restore should now be complete, and NICI reinitialized with the restored NICI files so you can access encrypted information. If you use roll-forward logging, you have prepared for any failures in the future by turning on roll-forward logging again after the restore and creating a new full backup as a baseline.

23 Auditing OpenText eDirectory Events

You can audit OpenText eDirectory events in one of the following ways:

- "Auditing with CEF" on page 579
- "Journal Event Caching" on page 600
- "LDAP Auditing" on page 601

Auditing with CEF

Common Event Format (CEF) provides a standard event format to facilitate the merging and analysis of audit information from multiple components at the distributed system level. The CEF format uses the Syslog message format as a transport mechanism.

CEF is an extensible text-based format that is designed to support multiple device types such as on-premise devices and cloud-based services. CEF events helps you easily understand the audit trails of heterogeneous applications.

Configuring OpenText eDirectory to use CEF provides the following benefits:

- Provides secured uniform audit services for a distributed system.
- CEF uses a standard message format that simplifies log management.
- The new event format seamlessly integrates with Sentinel.

This following section covers how to configure CEF with OpenText eDirectory:

- "Configuring CEF" on page 579

Configuring CEF

The OpenText eDirectory installation kit includes both a Linux and a Windows CEF client as part of its download package. The installation program for OpenText eDirectory installs the CEF packages on your operating system. The CEF package contains the following files:

- Linux
  - `novell-edirectory-xdaslog`
  - `novell-edirectory-xdaslog-conf`
  - `novell-edirectory-cefinstrument-9.2.0-0.x86_64.rpm`
- Windows
  - `cefauditds.dlm`
  - `xdaslog.dll`

This section provides the following information:

- "System Requirements" on page 580
- "Installing the OpenText Identity Console for CEF" on page 580
- "Configuring the CEF Property File" on page 581
- "Configuring CEF for Auditing" on page 587
- "Loading and Unloading the Modules" on page 590
- "Enabling CEF Event Caching" on page 590
- "Understanding CEF Event Types" on page 592
- "Using Collectors for CEF Events" on page 593
- "Understanding CEF Auditing Event Filtering" on page 594
- "CEF Implementation Schema" on page 595
- "CEF Events" on page 600

System Requirements

Installing and using the OpenText Audit Identity Console application requires OpenText Identity Console 1.5 at a minimum. See Planning to Install OpenText Identity Console for requirements and download instructions.

Installing the OpenText Identity Console for CEF

To install the OpenText Identity Console application:

1 Open OpenText Identity Console from a Web browser, using the following URL:

`https://ip_address_or_DNS:Port/identityconsole/`

where *ip_address_or_DNS* is the IP address or DNS name and port number of your OpenText Identity Console server.

For example:

`http://111.111.1.1:9000/identityconsole.html`

2 Log in to the OpenText Identity Console using tree user name and password.

If you want to access all the OpenText Identity Console features, you must login as an administrator to the tree. Only administrative user has full access to all the features. A non-administrative user can only access those roles for which rights are assigned.

For more information, see the OpenText Identity Console Administration Guide (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

Configuring the CEF Property File

The OpenText eDirectory media includes a sample properties file, `auditlogconfig.properties.template` file in the `configdir` (`n4u.server.configdir`) directory.

The following table lists the default location of the `auditlogconfig.properties` file on Linux and Windows operating systems.

Table 23-1 *CEF Configuration File*

Operating System	Location of the Property File
Linux	`/etc/opt/novell/eDirectory/conf/auditlogconfig.properties` For non-root installations, the CEF property file is located in the `conf` directory.
Windows	`<Install Path>/novell/nds/auditlogconfig.properties` The property file is usually in the OpenText eDirectory installation directory.

If you configure the property file and then upgrade your environment OpenText eDirectory 9.2 to any latest version, the installer does not replace the property file. Instead, the upgrade process updates the file (`auditlogconfig.properties`) to retain the customization.

You can configure CEF after installing OpenText Identity Console. The CEF configuration settings are stored in a simple text-based `auditlogconfig.properties` configuration file. You can customize the file based on your requirements.

The CEF auditlogconfig.properties file contains the following information:

Linux

```
# Set the level of the root logger to DEBUG and attach appenders.
#log4j.rootLogger=info, S, R

# Defines appender S to be a SyslogAppender.
#log4j.appender.S=org.apache.log4j.net.SyslogAppender

# Defines location of Syslog server.
#log4j.appender.S.Host=localhost
#log4j.appender.S.Port=port

# Specify protocol to be used (UDP/TCP/SSL)
#log4j.appender.S.Protocol=TCP

# Specify SSL certificate file for SSL connection.
# File path should be given with double backslash.
#log4j.appender.S.SSLCertFile=/etc/opt/novell/mycert.pem

# Minimum log-level allowed in syslog.
#log4j.appender.S.Threshold=INFO
```

```
# Defines the type of facility.
#log4j.appender.S.Facility=USER

# Defines caching for SyslogAppender.
# Inputs should be yes/no
#log4j.appender.S.CacheEnabled=no

# Cache location directory
# Directory should be available for creating cache files
#log4j.appender.S.CacheDir=/var/opt/novell/eDirectory

# Cache File Size
# Cache File size should be in the range of 50MB to 4000MB in limited
growth mode
# Cache File size should be set as 0MB to enable infinite growth of cache
file
#log4j.appender.S.CacheMaxFileSize=500MB

# Layout definition for appender Syslog S.
#log4j.appender.S.layout=org.apache.log4j.PatternLayout
#log4j.appender.S.layout.ConversionPattern=%c: %m%n

# Defines appender R to be a Rolling File Appender.
#log4j.appender.R=org.apache.log4j.RollingFileAppender

# Log file for appender R.
#log4j.appender.R.File=/var/opt/novell/eDirectory/log/cef-events.log

# Max size of log file for appender R.
#log4j.appender.R.MaxFileSize=100MB

# Set the maximum number of backup files to keep for appender R.
# Max can be 13. If set to zero, then there will be no backup files.
#log4j.appender.R.MaxBackupIndex=10

# Layout definition for appender Rolling log file R.
#log4j.appender.R.layout=org.apache.log4j.PatternLayout
#log4j.appender.R.layout.ConversionPattern=%d{MMM dd HH:mm:ss} %c %m%n
```

Windows

```
# Brief description for appenders and their options are provided.
# For detailed decriptions refer to log4cxx documentation.

# Set the level of the root logger to DEBUG and attach appenders.
#log4j.rootLogger=info, S, R

# Defines appender S to be a SyslogAppender.
#log4j.appender.S=org.apache.log4j.net.SyslogAppender

# Defines location of Syslog server.
#log4j.appender.S.Host=localhost
#log4j.appender.S.Port=port

# Specify protocol to be used (UDP/TCP/SSL).
#log4j.appender.S.Protocol=SSL

# Specify SSL certificate file for SSL connection.
```

```
# File path should be given with double backslash.
#log4j.appender.S.SSLCertFile=C:\\Novell\\mycert.pem

# Minimum log-level allowed in syslog.
#log4j.appender.S.Threshold=INFO

# Defines the type of facility.
#log4j.appender.S.Facility=USER

# Defines caching for SyslogAppender.
# Inputs should be yes/no
#log4j.appender.S.CacheEnabled=yes

# Cache location directory
# Directory should be available for creating cache files
#log4j.appender.S.CacheDir=C:\\NetIQ\\eDirectory

# Cache File Size
# Cache File size should be in the range of 50MB to 4000MB in limited
growth mode
# Cache File size should be set as 0MB to enable infinite growth of cache
file
#log4j.appender.S.CacheMaxFileSize=500MB

# Layout definition for appender Syslog S.
#log4j.appender.S.layout=org.apache.log4j.PatternLayout
#log4j.appender.S.layout.ConversionPattern=%c: %m%n

# Defines appender R to be a Rolling File Appender.
#log4j.appender.R=org.apache.log4j.RollingFileAppender

# Log file for appender R.
# File path should be given with double backslash.
#log4j.appender.R.File=C:\\cef-events.log

# Max size of log file for appender R.
#log4j.appender.R.MaxFileSize=100MB

# Set the maximum number of backup files to keep for appender R.
# Max can be 13. If set to zero, then there will be no backup files.
#log4j.appender.R.MaxBackupIndex=10

# Layout definition for appender Rolling log file R.
#log4j.appender.R.layout=org.apache.log4j.PatternLayout
#log4j.appender.R.layout.ConversionPattern=%d{MMM dd HH:mm:ss} %c %m%n
```

Before going through the content of the `auditlogconfig.properties` file, OpenText recommends reviewing the following considerations:

- The letters S and R specify Syslog Appender and Rolling File Appender respectively.
- The entries are not case sensitive.
- The entries can appear in any order.
- Empty lines in the file are valid.
- Any line that starts with a hash (#) is commented out.

The following table provides information about the settings in the `auditlogconfig.properties` file:

IMPORTANT: You must restart OpenText eDirectory after changing any configuration.

Setting	Description
log4j.rootLogger=**info, S, R**	Sets the level of the root logger to debug and attaches an appender named S or R, where S specifies a Syslog appender and R specifies a Rolling File appender. The available levels of root logger (in the same order of their priority) are: **trace**, **debug**, **info**, **warn**, **error** and **fatal**. By default, **info** will be specified which includes events from all other levels except **trace** and **debug**. If you want to send all the events to log, specify **TRACE** or **ALL** in this field.
log4j.appender.S=**org.apache.log4j.net.SyslogAppender**	Specifies the appender S to be a Syslog appender.
log4j.appender.S.Host=**localhost**	Specifies the location of the Syslog server where CEF events are logged. IFor example, log4j.appender.S.Host=192.168.0.1
log4j.appender.S.Port=**port**	The port at which the CEF connects to the Syslog server. The port supports values from 1 to 65535. If you specify an invalid value, the port defaults to 514. If the connection between CEF and the Syslog server fails, Identity Manager cannot log events until the connection is restored.
log4j.appender.S.Protocol=TCP	Specifies the protocol to use. For example, UDP, TCP, or SSL.
log4j.appender.S.SSLCertFile=**/etc/opt/novell/mycert.pem**	Specifies the SSL certificate file for the SSL connection. Use double backslashes to specify the path of the file. This is an optional setting.
log4j.appender.S.Threshold=**INFO**	Specifies the minimum log level allowed in the Syslog. The following log-levels are supported (in the same order of their priority): TRACE, DEBUG, INFO, WARN and ERROR. The minimum default log-level is INFO that includes all other log levels except TRACE and DEBUG. Which means, if you specify INFO as the minimum log-level, TRACE and DEBUG events will not be sent to the Syslog server. If you want to send all the events to the Syslog server, specify TRACE or ALL in this field.

Setting	Description
log4j.appender.S.Facility=**USER**	Specifies the type of facility. The facility is used to try to classify the message.Currently, USER facility is supported. These values may be specified as upper or lower case characters.
log4j.appender.S.layout=**org.apache.log4j.PatternLayout**	Layout setting for Syslog appender.
log4j.appender.S.layout.ConversionPattern=**%c : %p%m%n**	Layout setting for Syslog appender. For information about the conversion patters and their descriptions, see logging.apache.org.
log4j.appender.R=**org.apache.log4j.RollingFileAppender**	Specifies appender R to be a Rolling File appender.
log4j.appender.R.File=**/var/opt/novell/eDirectory/log/cef-events.log**	The location of the log file for a Rolling File appender.
log4j.appender.R.MaxFileSize=**100MB**	The maximum size, in MBs, of the log file for a Rolling File appender. Set this value to the maximum size that the client allows.
log4j.appender.R.MaxBackupIndex=**10**	Specify the maximum number of backup files for a Rolling File appender. The maximum number of the backup files can be 10. A 0 value means no backup files.
log4j.appender.R.layout=**org.apache.log4j.PatternLayout**	Layout setting for Rolling File appender.
log4j.appender.R.layout.ConversionPattern=**%d{MMM dd HH:mm:ss} %c : %p%m%n**	Layout setting for Rolling File appender. For information about the conversion patters and their descriptions, see logging.apache.org.

The following table lists examples with the date and time patterns interpreted in the U.S. The given date and time are 2012-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Table 23-2 *Date and Time Pattern Example*

Date and Time Pattern	Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2012.07.04 AD at 12:08:56 PDT
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02012.July.24 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 24 Jul 2012 12:08:56 -0700
"yyMMddHHmmssZ"	120724120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"	2012-07-04T12:08:56.235-0700

Enabling the Syslog Appender

You can use the Syslog appender to view the real time events. Additionally, a Syslog server offers better backup support in the event of a disaster.

To enable the Syslog appender, make the following changes in the `auditlogconfig.properties` file:

1. Change the following entry to S to attach a Syslog appender:

   ```
   log4j.rootLogger=info, S
   ```

2. Uncomment the following entries:

   ```
   log4j.appender.S=org.apache.log4j.net.SyslogAppender
   log4j.appender.S.Host=localhost
   log4j.appender.S.Port=port
   log4j.appender.S.Protocol=SSL
   log4j.appender.S.SSLCertFile=/etc/opt/novell/mycert.pem
   #log4j.appender.S.Threshold=INFO
   #log4j.appender.S.Facility=USER
   #log4j.appender.S.layout=org.apache.log4j.PatternLayout
   #log4j.appender.S.layout.ConversionPattern=%c: %m%n
   ```

3. Log into OpenText Identity Console and change the log events. For information about configuring CEF Events, see "Configuring the CEF Events for Auditing" on page 588.

NOTE: CEF caching using UDP protocol for `SyslogAppender` does not work.

Generating Certificate for Syslog SSL Connection

To generate a certificate for syslog connection:

1. Create the certificate by using the following OpenSSL command:

   ```
   openssl s_client -host LOG_SERVER  -port 1443 -showcerts
   ```

2. Specify the location of the certificate file that you created in the `/etc/opt/novell/eDirectory/conf/auditlogconfig.properties` file.

Enabling the Rolling File Appender

This File appender is preferred, if the auditing solution is limited to an individual server. Rolling file appender is more reliable as compared to the Syslog appender because it can store the events on your local file system and prevents loss of events. Also, it is easy to bring up this solution because the number of components to be setup are few and thus, is more suited for demonstrations.

NOTE: If you are using the File Connector for CEF, ensure that the conversion pattern for the Rolling File appender is similar to the Syslog appender as shown below:

```
log4j.appender.R.layout.ConversionPattern=%c: %m%n
```

To enable the Rolling File appender, make the following changes in the `auditlogconfig.properties` file:

1. Change the following entry to R to attach a Rolling File appender.

   ```
   log4j.rootLogger=info, R
   ```

2. Uncomment the following entries:

   ```
   log4j.appender.R=org.apache.log4j.RollingFileAppender

   log4j.appender.R.File=/var/opt/novell/eDirectory/log/cef-events.log

   log4j.appender.R.MaxFileSize=100MB

   log4j.appender.R.MaxBackupIndex=10

   log4j.appender.R.layout=org.apache.log4j.PatternLayout

   log4j.appender.R.layout.ConversionPattern=%d{MMM dd HH:mm:ss} %c %m%n
   ```

3. Select the desired event from OpenText Identity Console.

   For information about configuring CEF Events, see “Configuring the CEF Events for Auditing” on page 588.

Configuring CEF for Auditing

- “Using the OpenText Identity Console for Configuring CEF” on page 587
- “Configuring the CEF Events for Auditing” on page 588

Using the OpenText Identity Console for Configuring CEF

1. Open OpenText Identity Console from a Web browser using the following URL:

`https://ip_address_or_DNS/identityconsole/`

where *ip_address_or_DNS* is the IP address or DNS name and port number of your OpenText Identity Console server.

For example:

`http://111.111.1.1:9000/`

2 Log in using tree user name and password.

If you want to access all the OpenText Identity Console features, you must login as an administrator to the tree. Only administrative user has full access to all the features. A non-administrative user can only access those roles for which rights are assigned.

For more information, see the *OpenText IDentity Console Administration Guide (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/t48zlpkxbru1.html)*.

3 Select **Auditing**.

4 Select the required **NCP Server** for which you want to configure the auditing from the drop-down.

5 Specify the **Events Configuration** details if required.

6 Click **OK**.

The CEF Audit page is displayed. Continue with "Configuring CEF" on page 579.

NOTE: We recommend the administrator to disable the following attributes in the OpenText Audit Identity Console:

- Authoritative
- DirXML-DriverStorage
- Obituary
- Partition Status
- Replica
- Revision
- SAS:Login Secret

---

7 Click **Apply**. Click **OK**.

Configuring the CEF Events for Auditing

1 Log in to OpenText Identity Console using your user name and password.

2 Click **Auditing**.

3 Configure the CEF events.

- **Global:** You can select or clear the global settings for duplicate entries.
  - **Do Not Send Replicated Events:** Select this option to stop receiving duplicate events due to replication from other servers.
- **Log Event's Values:** The events are logged into a text file. Event values with more than 768 bytes in size are considered "large values." You can log events of any size.
  - **Log Large Values:** Select this option to log events that are more than 768 bytes in size.

- **Don’t Log Large Values:** Select this option to log events that are less than 768 byte in size.
- **Log Attributes Values:** Select this option to display the attribute values. This is applicable to **Add Value** and **Delete Value** events only.
- **Don’t Log Attribute Values:** Select this option to suppress the attribute values. This is applicable to **Add Value** and **Delete Value** events only. By default, this option is selected.
- **Log Encrypted Attribute Values:** Select this option to display the encrypted attribute values. This is applicable to **Add Value** and **Delete Value** events only.
- **Don’t Log Encrypted Attribute Values:** Select this option to suppress the encrypted attribute values. This is applicable to **Add Value** and **Delete Value** events only. By default, this option is selected.

NOTE: If the event size is more, the event value is truncated and saved to the log file.

- **Basic Events Configuration:** Specify the values for the following options based on the events required for your environment:

NOTE: Individual event categories under the basic events configuration section will be collapsed by default. You can expand each category to select individual events.

Options	Description
Security Events	Select the security events for which you want to log events. You can log events to add or delete member, to detect intruder, to change password and to authenticate users etc.
Object Events	Select the object events for which you want to log events. You can log events to create delete, rename, move and search objects.
Attribute Events	Select the attribute events for which you want to log events. You can log events to read and delete attributes and to add, delete and compare attribute value.
LDAP Events	Select the LDAP events for which you want to log events.
EBA Events	Select the EBA event for which you want to log events.

For information about OpenText eDirectory internal events mapped with the corresponding CEF events, see “Mapping OpenText eDirectory Events with CEF Events” on page 793.

NOTE: After modifying the event configuration, it takes up to 3 minutes for the configuration changes to take effect on the NCP Server. If you want the configuration changes to be effective immediately on the NCP server, you must unload and load the `cefauditds` module.

Loading and Unloading the Modules

After you configure the CEF events, run the following commands to load and unload the CEF modules:

To automatically load the `cefauditds` module whenever the `ndsd` server is started:

- **Linux**

  Add `cefauditds` to the `/etc/opt/novell/eDirectory/conf/ndsmodules.conf` file.

- **Windows**

  Run `ndscons.exe`, select `cefauditds` from the list of available modules, click **Startup**, and then select **Automatic** for Startup Type.

To manually load and unload the `cefauditds` module:

- **Linux**

  To load, run `ndstrace -c "load cefauditds"`.

  To unload, run `ndstrace -c "unload cefauditds"`.

- **Windows**

  To load, run `ndscons.exe`, select **cefauditds** from the list of available modules, then click **Start**.

  To unload, run `ndscons.exe`, select **cefauditds** from the list of available modules, then click **Stop**.

Enabling CEF Event Caching

OpenText eDirectory 9.2 allows you to optionally store CEF events locally on the agent in a Syslog Appender cache. With events cached, if the agent cannot communicate with the auditing server, the audit events generated are retained, ensuring that audit data is not lost. The agent then attempts to re-send the cached events when the agent computer can once again communicate with the auditing server.

CEF event caching is disabled by default. To enable event caching, complete the steps below.

1. On the agent computer, navigate to the location of the CEF property file. The `auditlogconfig.properties` file is located at `/etc/opt/novell/eDirectory/conf/auditlogconfig.properties` by default. For non-root installations, the CEF property file is located in the `conf` directory by default.
2. Use a text editor to open the `auditlogconfig.properties` file.
3. Within the property file, navigate to the `log4j.appender.S.CacheEnabled` property and change the property value to `yes`.
4. (Conditional) If you want to cache events in a specific directory, modify the value of the `log4j.appender.S.CacheDir` property to specify the directory path. If you specify a directory, ensure that the directory path is a valid location on the server. If the `log4j.appender.S.CacheDir` property is not set, the Syslog Appender logs cache events in the `dib` directory of that particular instance.

5 You can specify the custom file size for cache file in two different modes:

5a **Limited growth mode:** In limited growth mode, the minimum value for the cache file size should be configured at 50 MB, with a maximum value of 4 GB. Any value configured outside this range will set the cache file size value to 500MB by default. Once the cache file size limit reaches its maximum level, the cache file rollover starts. Cache file will not be truncated after all cached event logs are sent to Syslog server in this mode.

5b **Infinite growth mode:** To enable the infinite growth mode, set the Cache File size to 0MB. The cache file rollover does not happen in this mode. A truncation of cache file will happen once all cached events are sent to Syslog server in this mode. We recommend that you place the cache file in a separate partition or disk, to avoid impact to DIB growth due to cache file growth.

6 Save and close the `auditlogconfig.properties` file.

7 Reload the CEF module.

The CEF Caching Mechanism

The caching mechanism uses all of the following files under the Cache directory location:

- CEF-S-cache.log
- RolledOver-CEF-S-record
- OffsetWriter-CEF-S-record
- OffsetReader-CEF-S-record

The `CEF-S-cache.log` file is serviced by an Cache Writer and a Cache Reader. The Writer writes events into the Cache log when the connection to the Syslog server is lost. The `.OffsetWriter-CEF-S-record` file gets updated with the Writer Offset indicating the amount of CEF events logged into the Cache file. When the Writer reaches the maximum cache file size, a rollover happens to the beginning of the cache file which sets the flag in `.RolledOver-CEF-S-record` file to value 1 from 0. An appropriate message will be printed in the `ndsd.log` at this point indicating Cache Writer rollback. The Writer continues writing events into the Cache log and stops before the Reader's offset value. This is to ensure that the Writer does not overwrite the events which are yet to be sent to the Syslog server. Any event generated post this will be lost and an appropriate message indicating the full utilization of the Cache file will be printed in the `ndsd.log`.

The Reader reads the events from the cache file and sends them to the Syslog server when the connection is re-established. The `.OffsetReader-CEF-S-record` file gets updated with the Reader Offset indicating the amount of CEF events sent to the Syslog server from the Cache file. Reader position moves ahead giving room for the Writer to write more new events to cache. The Reader rolls back once it reaches the end of the file and the flag in `.RolledOver-CEF-S-record` file gets reverted to 0. An appropriate message will be printed in the `ndsd.log` at this point indicating Cache Reader rollback. Read and send continues until the Reader offset matches the Writer offset. This procedure sends all Cached data to the Syslog server. After this process completes, no further update happens to the Cache file as events will be sent directly to the Syslog server.

NOTE: Since rollover does not happen in Infinite growth mode, the Cache file gets truncated after all Cached events are sent to the Syslog server. All offsets will also be reset once the truncation of the Cache file completes.

Understanding CEF Event Types

You can configure CEF to log events in the following categories:

- Security
- Objects
- Attributes
- LDAP
- EBA

You can audit the following default set of event types:

Category	Event Type
Security	◆ ACL Changed ◆ Add Member ◆ Delete Member ◆ Intruder Detected ◆ Login Disabled ◆ Login Enabled ◆ Login ◆ Change Security Equals ◆ Audit Config ◆ Change Password ◆ Account Unlock ◆ Logout ◆ Connection ◆ Impersonate ◆ Authenticate ◆ Verify Password ◆ Change Login Config ◆ Query Credential
Objects	◆ Create Object ◆ Delete Object ◆ Rename Object ◆ Move Object ◆ DSA Read ◆ Search

Category	Event Type
Attributes	◆ Read Attribute ◆ Delete Attribute ◆ Add Value ◆ Delete Value ◆ Compare Attribute Value
LDAP	◆ LDAP Bind ◆ LDAP Bind Response ◆ LDAP Unbind ◆ LDAP Connection ◆ LDAP Search ◆ LDAP Search Response ◆ LDAP Search Entry Response ◆ LDAP Add ◆ LDAP Add Response ◆ LDAP Compare ◆ LDAP Compare Response ◆ LDAP Modify ◆ LDAP Modify Response ◆ LDAP Delete ◆ LDAP Delete Response ◆ LDAP Modify DN ◆ LDAP Modify DN Response ◆ LDAP Abandon ◆ LDAP Extended Operation ◆ LDAP System Extended Operation ◆ LDAP Extended Operation Response ◆ Modify LDAP Server Configuration ◆ Unknown LDAP Operation ◆ LDAP Password Modify
EBA	◆ Modify Service Config

NOTE: The event *Modify LDAP Server Configuration* is published when the LDAP server is refreshed through an LDAP client request.

Using Collectors for CEF Events

For more information about using collectors for collecting CEF events, see the OpenText eDirectory Collector documentation at the Sentinel Plug-ins page.

Understanding CEF Auditing Event Filtering

Using filters and event notifications, CEF is capable of reporting when a specific type of event occurs, or when it does not occur. You can also filter events for one or more specific object classes or attributes, depending on the event type. CEF evaluates all the generated events against the configured filters on the OpenText eDirectory server and logs only the events matching to those filters.

This section provides the information you need to configure your system filters and notifications.

- "Filtering CEF Object Events" on page 594
- "Filtering CEF Attribute Events" on page 594
- "Filtering OpenText eDirectory Events With Exclusion Filter" on page 595

Filtering CEF Object Events

You can configure filtering for Objects to look for only a specific event or events. For example, if you want to be notified when someone creates a user account in OpenText eDirectory, you can create a filter selecting the User Object class to log events for creating a new user object.

To configure accounts filtering, click the Object Events link, select the class, and then click **OK** to exit the application.

To configure filters for Account Management events:

1. In OpenText Identity Console home page, navigate to **Auditing**.
2. Select the **NCP Server** you want to monitor from the drop-down menu, and then click **OK**.
3. Click **Object Events** tile.
4. Select the required object event to be configured.
5. Click **Apply** and click **OK** to successfully modify the auditing details.

Using the configured filter, CEF audit module checks all generated events for the selected object classes and attributes and logs those events.

Filtering CEF Attribute Events

Click the **Attribute Events** link to configure filtering for the Attribute Events. For example, if you want to be notified when someone adds a new attribute value in OpenText eDirectory, you can create a filter to log events for adding a new value.

To configure filtering for Trust Management Events:

1. In OpenText Identity Console, navigate to **eDirectory Auditing**.
2. Select the NCP Server you want to monitor, and then click **OK**.
3. Click **Attribute Events**.

   The **Attribute Filtering** window appears.
4. In the **Available Classes** list, select object classes for which you want to collect events, then click the right arrow to move them to the **Selected Classes** list. By default **dynamicGroup**, **dynamicGroupAux**, **Group**, **LDAP Group** and **Organizational Role** object classes are selected

5 In the **Available Attributes(s)** list, select any number of attributes for the selected object classes. Select the attribute and click the right arrow to add the attribute to the selected list of attributes.

NOTE: If you select an object class, then all the Attribute Events for all attributes on that object class are selected. In this case, you will get all the Attribute Events for the all attributes on the selected object classes.

---

6 Click **OK**.

With the filter configured, CEF audit module checks the generated events for all the selected object classes and attributes and logs those events.

Filtering OpenText eDirectory Events With Exclusion Filter

Click the **Exclusion Filter** link to configure filtering for those object classes and attributes for which you do not want an event to be generated. You can select object classes and attributes.

To configure filtering for unwanted OpenText eDirectory Events:

1 On the OpenText Identity Console home page, click **Auditing** tile.

2 Select the NCP Server you want to monitor, and then click **OK**.

3 Click **Advanced Setting** drop down > **Exclusion Filter**.

The CEF Exclusion Filtering window appears.

4 In the **Available Object Classes** list, select object classes for which you do not want to collect events, then click the right arrow to move them to the **Selected Object Classes** list.

5 In the **Available Attribute(s)** list, select any number of attributes. Select the attribute and click the right arrow to add the attribute to the selected list of attributes.

6 Click **OK**.

Using the configured filter, CEF audit module stops generating events for all the selected object classes and attributes.

CEF Implementation Schema

This document defines the CEF protocol and provides details on how to implement the standard. It details the header and predefined extensions used within the standard.

Using CEF with Syslog

CEF uses syslog as a transport mechanism. It uses the following format, comprised of a syslog prefix, a header and an extension, as shown below:

```
Jan 18 11:07:53 host CEF:Version|Device Vendor|Device Product|Device
Version|Device Event Class ID|Name|Severity|[Extension]
```

The `CEF:Version` portion of the message is a mandatory header. The remainder of the message is formatted using fields delimited by a pipe ("|") character. All of these remaining fields should be present and are defined under "CEF Field Definitions" on page 596.

The extension portion of the message is a placeholder for additional fields, but is not mandatory. Any additional fields are logged as key-value pairs. For more information, see "CEF Field Definitions" on page 596.

CEF Field Definitions

These fields in the schema are the CEF fields defined specifically for audit events. Some or all of these fields may also be relevant to other types of event, but information of this sort is required for auditing services.

Table 23-3 *CEF Field Definitions*

CEF Field	Description
Device Product	Device Product is a string that uniquely identifies the sending device. Two products cannot have the same device-product pair. The administrator must ensure to assign unique name to every device-product pair.
Device Version	Device Version is a string that uniquely identifies the sending device. Two products cannot have the same device-version pair. The administrator must ensure to assign unique name to every device-version pair.
Device Vendor	Device Vendor is a string that uniquely identifies the sending device. Two products cannot have the same device-vendor pair. The administrator must ensure to assign unique name to every device-vendor pair.
Device Event Class ID	Device Event Class ID is a unique identifier per event-type. This can be a string or an integer. Device Event Class ID identifies the type of event reported. In the intrusion detection system (IDS) world, each signature or rule that detects certain activity has a unique Device Event Class ID assigned. This is a requirement for other types of devices as well, and helps correlation engines process the events. This is also known as Signature ID.
Severity	This is a string or integer and reflects the importance of the event. The valid string values are **Unknown**, **Low**, **Medium**, **High**, and **Very-High**. The valid integer values are 0-3=Low, 4-6=Medium, 7-8=High, and 9-10=Very-High. From eDirectory 9.2 SP2 onwards, the following mapping of values and integers are followed: ◆ 1 - TRACE ◆ 2 - DEBUG ◆ 4 - INFO ◆ 6 - WARN ◆ 8 - ERROR
Version	This is an integer and identifies the version of the CEF format. Event consumers use this information to determine what the following fields represent. The current CEF version is 0.
Device Address	Identifies the device address that an event refers to in an IP network. The format is an IPv4 address.
c6a1	One of four IPV6 address fields available to map fields that do not apply to any other in this dictionary.

CEF Field	Description
dvchost	The format should be a fully qualified domain name associated with the device node, when a node is available. For example, `host.domain.com` or `host`.
rt	The time at which the event related to the activity was received. The format is MMM dd yyyy HH:mm:ss or milliseconds since epoch.
dtz	The timezone for the device generating the event..
sourceServiceName	The service which is responsible for generating this event.
sproc	The name of the event's source process.
src	Identifies the source that an event refers to in an IP network. The format is an IPv4 address. For example, `192.168.10.1`.
spt	This is the source port number. The valid port numbers are 0 to 65535.
shost	Identifies the source that an event refers to in an IP network. The format should be a fully qualified domain name associated with the source node, when a node is available. For example, Examples: `host.domain.com` or `host`.
suser	Identifies the source user by name. Email addresses are also mapped into the UserName fields. The sender is a candidate to put into `sourceUserName`.
dst	Identifies the destination address that the event refers to in an IP network. The format is an IPv4 address. For example, `192.168.10.1`.
duser	Identifies the destination user by name. This is the user associated with the event's destination. Email addresses are often mapped into the UserName fields. The recipient is a candidate to put into `destinationUserName`.
cn1	One of three number fields available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomNumber1`.
cn2	One of three number fields available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomNumber2`.
cn3	One of three number fields available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomNumber3`.
cn1Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomNumber1Label.`
cn2Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomNumber2Label`.
cn3Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomNumber3Label`
cs1	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString1`.

CEF Field	Description
cs2	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString2`.
cs3	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString3`.
cs4	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString4`.
cs5	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString5`.
cs6	One of six strings available to map fields that do not apply to any other in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. Also known as `deviceCustomString6`.
cs1Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString1Label`.
cs2Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString2Label`.
cs3Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString3Label`.
cs4Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString4Label`.
cs5Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString5Label`.
cs6Label	All custom fields have a corresponding label field where the field itself can be described. Each of the fields is a string describing the purpose of this field. Also known as `deviceCustomString6Label`.
flexString1	One of two string fields available to map String data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary.
flexString2	One of two string fields available to map String data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary.

CEF Field	Description
flexString1Label	One of two string fields available to map String data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary.
flexString2Label	One of two string fields available to map String data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary.
flexNumber1	One of two number fields available to map Long data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary
flexNumber2	One of two number fields available to map Long data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary
flexNumber1Label	One of two number fields available to map Long data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary
flexNumber2Label	One of two number fields available to map Long data that does not apply to any other field in this dictionary. Use sparingly and seek a more specific, dictionary supplied field when possible. These fields are typically reserved for customer use and should not be set by vendors unless necessary
cat	Represents the category assigned by the originating device. Devices oftentimes use their own categorization schema to classify events. Also known as `deviceEventCategory`. For example, `/Monitor/Disk/Read`.
reason	The reason for which an audit event was generated. For example `Bad password` or `Unknown User`. This could also be an error or return code.
outcome	Displays the outcome, usually as `success` or `failure`. Also known as `eventOutcome`.
msg	Description of the event

Example of an Event

An example event is given below:

```
Apr 15 19:43:37 eDirectory
CEF:0|NetIQ|eDirectory|9.2.2|CEF0B035D|AUTHENTICATE|4|dvc=164.99.179.219
dvchost=WIN-ADKHVNQHFCR rt=1586960017190 dtz=India Standard Time
sourceServiceName=CN\=WIN19,O\=novell sproc=eDirectory#DS
src=164.99.179.219 spt=0 suser=CN\=admin,O\=novell
duser=CN\=admin,O\=novell cs2Label=Class Name cs2=User cs3Label=Tree Name
cs3=WIN19_tree cs4Label=Correlation ID cs4=eDirectory#24# cs6Label=Server
Name cs6=CN\=WIN19,O\=novell flexString2Label=SubEvent
flexString2=DSE_AUTHENTICATE flexNumber2Label=Grouping flexNumber2=2131
cat=Security reason=0 outcome=Success msg=CN\=admin,O\=novell (Class:
User) authenticated from  to server CN\=WIN19,O\=novell : Success
```

CEF Events

For more information about CEF events, see Appendix G, “Mapping OpenText eDirectory Events with CEF Events,” on page 793.

Journal Event Caching

OpenText eDirectory has an event system where event consumers can register for events and consume them when they occur. An event handler can be registered as Worker, Inline, or Journal. The Journal Event queue is expected to report the events in the same order as they occur. With the current Journal Event system, the journal event queue is maintained in memory. If the consumers of the events are slow, or the rate at which events occur is more than the rate at which they can be processed, the Journal queue starts growing. This results in the memory growth of the ndsd process.

The Journal Event system is modified to allow you to use a combination of memory and disk to maintain events in a queue. This reduces the drastic growth in memory of the ndsd process.

Some instances where the events can cause memory growth are: *ndstrace enabled* or *auditing enabled*. You can control memory growth by enabling Event system caching.

Configuring Event System Caching

You must set the following environment variables for event system caching:

- NDSD_EVENT_DISK_CACHE

  This variable controls the use of new event system. By default, the new event system is disabled. To enable the new event system, export this variable with a value *true* or *1*.

- (Optional) NDSD_EVENT_DISK_CACHE_DIR

  This variable specifies the temporary location where event files are created. Under the specified directory, another sub-directory `cdir` is created, if it is not already present. At start up, all files inside the sub directory are cleaned up. We recommend that you set the caching directory in a different disk partition, and not in the same partition as that of DIB.

  In Linux, if NDSD_EVENT_DISK_CACHE_DIR is not specified or the specified directory is not accessible, ndsd uses `vardir` as the caching directory. By default, the value of `vardir /var/opt/novell/eDirectory/data/`.

  In Windows, if this variable is not specified or the specified directory is not accessible, dhost uses the `DIBFiles` directory.

NOTE: Ensure that there is sufficient disk space available in the caching directory because ndsd/dhost can quickly consume several GBs of disk space.

LDAP Auditing

Auditing is one of the primary functionalities that an administrator will be interested in when evaluating a directory. The OpenText eDirectory event mechanism facilitates OpenText eDirectory auditing. Because the applications are largely adopting the LDAP protocol for accessing directories, the requirement of auditing LDAP operations is becoming prevalent.

This chapter consists of the following sections:

- "Need for LDAP Auditing" on page 601
- "Using LDAP Auditing" on page 601
- "For More Information" on page 602

Need for LDAP Auditing

This event mechanism was noticeably absent in the existing OpenText eDirectory LDAP server that could not provide sufficient LDAP information. Though NDS event system produced events for all OpenText eDirectory operations, most of this information was insufficient or irrelevant for an application to audit the LDAP server. Information that covers protocol and bind details, network address, authentication methods, authentication types, LDAP search and transaction details, and so on, that is vital for auditing an LDAP server, was not available with the NDS events. Applications developers found it difficult to write to LDAP audit applications based on these events

LDAP is an important interface of OpenText eDirectory. To provide a mechanism for applications to audit OpenText eDirectory LDAP server, OpenText eDirectory includes an LDAP event subsystem. This subsystem generates LDAP specific events with all the relevant information for an application to audit an LDAP server. This is known as LDAP Auditing.

Using LDAP Auditing

LDAP Auditing enables the applications to monitor/audit LDAP operations such as Add, Modify, Search, and so on, and fetches useful information from the LDAP server such as the connection information, the client IP to which the server was connected at the time of LDAP operation, the message ID, the result code of the operation, and so on.

LDAP Auditing can be exercised through the LDAP Libraries for C (https://www.microfocus.com/documentation/edirectory-developer-documentation/ldap-libraries-for-c/), that provides the client side interface for this feature through new LDAP structures and events.

For More Information

For more information on LDAP Auditing Events, see the following documentation:

- NDK LDAP Tools (https://www.microfocus.com/documentation/edirectory-developer-documentation/ldap-classes-for-java/docs/jldapenu.pdf) in the LDAP Libraries for C documentation.
- For information on LDAP tools, see LDAP Libraries for C (https://www.microfocus.com/documentation/edirectory-developer-documentation/ldap-libraries-for-c/).

24 Understanding OpenText eDirectory's Authentication Framework

This section provides an overview of NMAS. NMAS is automatically installed with OpenText eDirectory. For more information about supported platforms and installation instructions, see the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

- "NMAS Functionality" on page 603
- "NMAS Software" on page 607
- "Managing Login and Post-Login Methods and Sequences" on page 609
- "Using NMAS to Log In to the Network" on page 616
- "History of OpenText Passwords" on page 617
- "NMAS HOTP Based Login" on page 618
- "Other Administrative Tasks" on page 624
- "Security Considerations" on page 629

NMAS Functionality

NMAS is designed to help you protect information on your network. In addition to the Password Management tool, NMAS brings together ways of authenticating to OpenText eDirectory networks. This helps to ensure that the people accessing your network resources are who they say they are.

NMAS employs three different phases of operation during a user's session on a workstation with respect to authentication devices. These phases are as follows:

1. User Identification Phase (who are you?)
2. Authentication (Login) Phase (prove who you say you are)
3. Device Removal Detection Phase (are you still there?)

All three of these phases of operation are completely independent. Authentication devices can be used in each phase, but the same device need not be used each time.

User Identification Phase

This is the process of gathering the username. Also provided in this phase are the tree name, the user's context, the server name, and the name of the NMAS sequence to be used during the Authentication phase. This authentication information can be obtained from an authentication device, or it can be entered manually by the user.

Authentication (Login) Phase

- "Password Authentication" on page 604
- "Physical Device Authentication" on page 605
- "Biometric Authentication" on page 605

NMAS uses three different approaches to logging in to the network called **login factors**. These login factors describe different items or qualities a user can use to authenticate to the network:

- Password Authentication (something you know)
- Physical Device Authentication (something you have)
- Biometric Authentication (something you are)

For more information on these login factors, see "Login and Post-Login Methods and Sequences" on page 606.

Password Authentication

Passwords (something you know) are important methods for authenticating to networks. NMAS provides several password authentication options:

- **NDS Login Method:** The NDS password is stored in a hash form that is non-reversible and only the NDS system can make use of this password. This option uses the Universal Password if it is enabled and set.
- **SCRAM Login Method:** The Salted Challenge Response Authentication Mechanism (SCRAM) uses PBKDF2 hash of passwords to authenticate users when they try to login to the OpenText eDirectory servers (RFC 5802). For more information, see "Understanding Non-Reversible Password Storage" on page 710.

  NOTE: While creating a new OpenText eDirectory 9.2 tree, the SCRAM method will be installed automatically. While upgrading any previous version of OpenText eDirectory tree to 9.2, you must manually install the SCRAM method. For more information, see "Ways of Installing a Login Method" on page 610.

- **Simple password:** The simple password allows administrators to import users and passwords (clear text and hashed) from foreign LDAP directories. This option uses the Universal Password if it is enabled and set.
- **Digest-MD5 SASL:** Digest-MD5 SASL provides the IETF standard DIGEST-MD5 SASL mechanism that validates a password hashed by the MD5 algorithm to be used for a LDAP SASL bind. This option will use the Universal Password if it is enabled and set.
- **Challenge/Response:** Challenge/Response provides a way for a user to prove his or her identity using one or more responses to pre-configured challenge questions.

Universal Password is a way to simplify the integration and management of different password and authentication systems into a coherent network. For more information on Universal Password, see Chapter 26, "Managing Passwords," on page 707.

Physical Device Authentication

OpenText developers and third-party authentication developers have written authentication modules for NMAS for several types of physical devices (something you have):

NOTE: NMAS uses the word to refer to all physical device authentication methods (smart cards with certificates, one-time password (OTP) devices, proximity cards, etc.).

- **Smart card:** A smart card is a plastic card, about the size of a credit card, or a USB device that includes an embedded, programmable microchip that can store data and perform cryptographic functions. With NMAS, a smart card can be used to establish an identity when authenticating to OpenText eDirectory.

  OpenText provides the OpenText Enhanced Smart Card login method for the use of smart cards. The OpenText Enhanced Smart Card login method is provided as part of the Identity Assurance Client. For more information, see the *OpenText Enhanced Smart Card Method 3.0 Installation and Administration Guide*.
- **One-Time Password (OTP) device:** An OTP device is a hand-held hardware device that generates a one-time password to authenticate its owner.
- **Proximity card:** A proximity card is a card worn by a person. This technology locks and unlocks a person's workstation based on the card's proximity to the workstation.

  OpenText provides the pcProx login method, which supports RFID proximity cards. The pcProx login method is provided as part of the OpenText SecureLogin product. For more information, see *NMAS Login Method and Login ID Snap-In for pcProx*.

Biometric Authentication

Biometrics is the science and technology of measuring and statistically analyzing human body characteristics (something you are). Biometric methods are provided by third-party companies for use with NMAS.

Biometric authentication requires readers or scanning devices, software that converts the scanned information into digital form, and a database or directory that stores the biometric data for comparison with entered biometric data.

In converting the biometric input, the software identifies specific points of data as match points. The match points are processed by using an algorithm to create a value that can be compared with biometric data scanned when a user tries to gain access.

Some examples of biometric authentication include scans of fingerprints, retinas, irises, and facial features. Biometrics can also include, handwriting, typing patterns, voice recognition, etc.

Device Removal Detection Phase

The user's session enters this phase after login is complete. Two methods are available:

- The Secure Workstation method, which is available with OpenText SecureLogin. The user's session can be terminated when an authentication device (such as a smart card) is removed. This device need not be used in any of the other phases.

For more information on the Secure Workstation method, see the OpenText SecureLogin Administration Guide (https://www.microfocus.com/documentation/securelogin/9.2/administration_guide/front.html).

- The OpenText Enhanced Smart Card login method also provides smart card removal detection. For more information on the OpenText Enhanced Smart Card login method, see the *OpenText Enhanced Smart Card Method Installation Guide*.

Login and Post-Login Methods and Sequences

A **login method** is a specific implementation of a login factor. NMAS provides multiple login methods to choose from based on the three login factors (password, physical device or, and biometric authentication).

A *post-login method* is a security process that is executed after a user has authenticated to OpenText eDirectory. For example, one post-login method is the OpenText Secure Workstation method (available with OpenText SecureLogin), which requires the user to provide credentials in order to access the computer after the workstation is locked.

NMAS software includes support for a number of login and post-login methods from OpenText and from third-party authentication developers. Additional hardware might be required, depending on the login method. Refer to the third-party product's documentation for more information.

After you have decided upon and installed a method, you need to assign it to a login sequence in order for it to be used. A *login sequence* is an ordered set of one or more methods. Users log in to the network by using these defined login sequences. If the sequence contains more than one method, the methods are presented to the user in the order specified. Login methods are presented first, followed by post-login methods.

Both And and Or login sequences exist with NMAS. An And login sequence requires all of the login methods in the sequence to complete successfully. An Or login sequence requires only one of the login methods in the sequence to complete successfully. An example of an Or login sequence is to allow users to use the same login sequence to login to workstations with different authentication devices.

Security Object Caching

The security container is created off the root partition when the first server is installed in the tree and holds information such as global data, security policies, and keys.

After universal password was introduced, whenever a user logged into OpenText eDirectory through NMAS, NMAS accessed the information in the security container to authenticate the login. When the partition having the security container was not present locally, NMAS accessed the server, which had this partition. This had an adverse impact on the performance of NMAS authentication. The situation was worse in the scenarios where the server containing the partition having the security container had to be accessed over WAN links.

To resolve this, the security container data is cached onto the local server. Therefore, NMAS does not need to access the security container located on a different machine whenever a user logs in, it can easily access it locally. This increases the performance. Adding the partition having security container to local server improves the performance, but it might not be feasible in scenarios where there are too many servers.

If the actual data in the security container changes on the server containing the security container partition, the local cache is refreshed by a background process called backlinker. By default, backlinker runs every thirteen hours and it pulls the modified data from remote server. In case, the data needs to be synchronized immediately, you can schedule backlinker on the local server either through OpenText iMonitor, ndstrace on Linux, or ndscons on Windows. For more information, refer to the OpenText iMonitor online help or the ndstrace manpage.

The security object caching feature is enabled by default. If you do not want backlinker to cache any data, remove `CachedAttrsOnExtRef` from the NCP server object.

NMAS Software

NMAS is included as a bundled product with OpenText eDirectory. The software image includes the following:

- NMAS server software
- Login methods software
- Support for multiple login methods per login sequence
- Support for graded authentication
- Universal Password

NMAS client software is available with the OpenText Client for Windows and with OpenText SecureLogin.

- "Server and Client Software Installation" on page 607
- "Login Method Software and Partners" on page 608
- "Universal Password" on page 608
- "OpenText Identity Console Management" on page 609

Server and Client Software Installation

NMAS server-side software is installed with OpenText eDirectory by default. NMAS client-side software must be installed on each client workstation that will access the network using the NMAS login methods. After installation, you can manage NMAS by using **Authentication Management** tile in OpenText identity Console.

The NMAS client software now ships with the OpenText Client. For more information, refer to the OpenText Client for Windows documentation.

During the installation, NMAS extends the OpenText eDirectory schema and creates new objects in the Security container in the OpenText eDirectory tree. These new objects are the Authorized Login Methods container, the Authorized Post-Login Methods container, the Security Policy object, and the Login Policy object. All login methods are stored and managed in the Authorized Login Methods container. All post-login methods are stored and managed in the Authorized Post-Login Methods container.

Login Method Software and Partners

- "Software and Partners" on page 608
- "Installing a Login Method" on page 608

Software and Partners

Several currently supported login methods are available on the NMAS software image.

NMAS software includes support for a number of login methods from third-party authentication developers. Refer to the OpenText Partners Web site (https://www.opentext.com/partners/find-an-opentext-partner) for a list of OpenText partners.

Each partner that develops login methods for NMAS addresses network authentication with unique product features and characteristics. Therefore, each login method varies in its actual security properties.

OpenText has not evaluated the security methodologies of these partner products, so although these products might have qualified for the OpenText Yes, Tested & Approved or OpenText Directory Enabled logos, those logos relate to general product interoperability only.

We encourage you to carefully investigate each partner's product features to determine which product will best meet your security needs. Also note that some login methods require additional hardware and software not included with the NMAS product.

Installing a Login Method

NMAS login methods can be installed by using the following:

- `nmasinst` (available on all OpenText eDirectory platforms), which requires OpenText eDirectory to be installed
- **Authentication Management** tile in OpenText Identity Console.

For more information on installing a login method, see "Ways of Installing a Login Method" on page 610

Universal Password

Universal Password is a way to simplify the integration and management of different password and authentication systems into a coherent network. It provides one password for all access to OpenText eDirectory, enables the use of extended characters in passwords, enables advanced password policy enforcement, and allows synchronization of passwords from OpenText eDirectory to other systems.

For more information on Universal Password, see Chapter 26, "Managing Passwords," on page 707.

OpenText Identity Console Management

You can manage NMAS by using **Authentication Management** tile in OpenText Identity Console. It is a Web-based utility for managing OpenText eDirectory. Specific property pages in OpenText Identity Console let you manage login methods, login sequences, enrollment, and graded authentication.

By default, NMAS installs the standard NDS password login method. Additional login methods can be installed by using Authentication Management in OpenText Identity Console, and a wizard launched from the Authorized Login Methods container using the Create New Object option. Post-login methods can be installed using a wizard launched from the Authorized Post-Login Methods container using the Create New Object option.

For more information about installing login methods, see "Ways of Installing a Login Method" on page 610.

Managing Login and Post-Login Methods and Sequences

This section describes how to install, set up, and configure login and post-login methods and sequences for NMAS.

NMAS provides multiple login methods to choose from, based on the three login factors (password, physical device or, and biometric authentication).

NMAS includes support for a number of login and post-login methods from OpenText and from third-party authentication developers. Some methods require additional hardware and software. Make sure that you have all of the necessary hardware and software for the methods you will use.

NMAS includes several login methods in the software build. Other login methods are available from third-party vendors.

See the OpenText Partners Web site for a list of OpenText eDirectory partners. Some partners develop third-party login methods.

- "Ways of Installing a Login Method" on page 610
- "Updating Login and Post-Login Methods" on page 611
- "Managing Login Sequences" on page 611
- "Authorizing Login Sequences for Users" on page 613
- "Setting Default Login Sequences" on page 614
- "Deleting a Login Method" on page 614
- "Deleting a Login Sequence" on page 615

Ways of Installing a Login Method

You have three ways of installing a login method for use in OpenText eDirectory:

- nmasinst utility (Linux and Windows), which allows you to install login methods into OpenText eDirectory.
- OpenText Identity Console (Linux and Windows), which allows you to install login and post-login methods into OpenText eDirectory.

- "Using the nmasinst Utility to Install a Login Method" on page 610
- "Using OpenText Identity Console to Install a Login or Post-Login Method" on page 610

Using the nmasinst Utility to Install a Login Method

From the server console command line, enter:

```
nmasinst -addmethod admin.context treename config.txt_path [-h
hostname[:port]] [-w password|file:<filename>|env:<environment_variable>]
[-checkversion] [-d]
```

- *admin.context*: The admin name and context.
- *treename*: The name of the OpenText eDirectory tree where you are installing the login method.
- *config.txt_ path* - The complete or relative path to the `config.txt` file of the login method. A `config.txt` file is provided with each login method.
- `[-h` *`hostname`*`[:`*`port`*`]]`: (Optional) The hostname and port of the server. Use this if OpenText eDirectory is not running on the default port. You can also specify the IP address. OpenText eDirectory 9.2 supports both IPv4 and IPv6 addresses. For example:
  - **IPv4:** `-h 127.0.0.1:8443`
  - **IPv6:** `-h [2001:db8::6]:8443`
- `[-w` *`password`*`|file:<`*`filename`*`>|env:<`*`environment_variable`*`>]`: This option allows you to specify the password using one of the following methods:
  - On the command line. For example: `-w n`
  - Through a file. For example: `-w file:/tmp/passwd`
  - Through an environment variable. For example: `-w env:PASSWD`
- `[-checkversion]`: This option reports an error if the installed method version is the same or newer than the method version being installed.
- `[-d]`: Delete methods for unsupported platforms.

If the login method already exists, nmasinst updates it.

Using OpenText Identity Console to Install a Login or Post-Login Method

1. Launch OpenText identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.

3 On the OpenText Identity Console home page > **Authentication Management** > **Login Methods and Sequences**.

4 Click **Login Methods** tab.

5 Click **install Login methods** +

6 Browse for and select the login method (`.zip`) file you want to install, then click **Next**.

7 Follow the installation wizard to completion.

Updating Login and Post-Login Methods

When a login method vendor provides an update for a login or post-login method, you can update the method by doing the following:

- "Using the nmasinst Utility to Update a Login Method" on page 611
- "Using OpenText Identity Console to Update a Login Method" on page 611

Using the nmasinst Utility to Update a Login Method

Use the same procedure you used to install a login method with the nmasinst utility (see "Using the nmasinst Utility to Update a Login Method" on page 611). Include the path to the new `config.txt` file and the login method is updated.

Using OpenText Identity Console to Update a Login Method

1 Launch OpenText Identity Console.

2 Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.

3 On the **Authentication Management** tile > **Login Methods**.

4 Select the login method you want to update.

5 On the login method property page, click **Update Method**.

6 Follow the update wizard to completion.

Managing Login Sequences

When you install a login, you are asked if you want to create a login sequence that uses only the login method you are installing. If you answer yes, a login sequence is created for you that contains just the one login method.

You can also manually create and manage login sequences. After login and post-login methods are installed, you can view, add, modify, or delete login sequences by using OpenText Identity Console. Login sequences are not created when methods are modified or updated.

In NMAS, you can set up multiple login and post-login methods per sequence. You must have at least one login method selected to be able to select a post-login method.

When multiple methods are selected for a sequence, they are executed in the order they are listed. Login methods are executed first, then post-login methods.

A login sequence can be an And or an Or sequence. An And sequence is successful if all of the login methods successfully validate the identity of the user. An Or sequence only requires that one of the login methods validate the identity of the user for the login to be successful.

The post-login methods are only executed if the login is successful, regardless of the And/Or relationship.

After a sequence is created, you can authorize users to use the new sequence to log in to OpenText eDirectory.

- "Creating a New Login Sequence by Using OpenText Identity Console" on page 612
- "Modifying a Login Sequence" on page 612
- "Deleting a Login Sequence" on page 613

Creating a New Login Sequence by Using OpenText Identity Console

1. Launch OpenText Identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.
3. Click **Authentication Management** > **Login Method Sequences**.
4. Click **Create** + and specify a name for the new login sequence.

   All available methods are listed under **Available Login Methods** and **Available Post-Login Methods**.
5. Select the **Sequence Type** from the list.

   If you select *And*, a user must log in using every login method that makes up the login sequence. If you select *Or*, the user only needs to log in using one of the login methods that makes up the login sequence.
6. Use the arrows to add each desired method to the sequence.

   If you are using multiple methods, use the vertical arrows to change the execution order.

   The **Sequence Grade** field displays the grade for the login sequence. For And sequences, the sequence grade is the union of the grades of the login methods. For Or sequences, the sequence grade is the intersection of the method grades.
7. Click **Create** to save the login sequence.

Modifying a Login Sequence

1. Launch OpenText Identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.
3. Go to the **Authentication Management** tile > **Login Method Sequences**.
4. Click **Login Method Sequences**. The sequence **Name, Grade, Authorized** and **Default** lists are displayed, and the **Login Methods** and **Post-Login methods** are listed.

5 Select an action:

5a Click + to create login method Sequence. Specify the **Name** and **Sequence Type** drop down value to be created. All the available methods appear in the **Available Login Methods** and **Available Post-Login Methods** lists.

5b Click  to modify the existing login method Sequence.

NOTE: You must have at least one login method selected in order to select a post-login method.

---

6 To change the sequence order of the **Login Methods**, use the up-arrow and down-arrow.

7 To change the **Sequence Type**, use the drop-down list next to Sequence Type.

8 Click **Save** or click **Cancel** to **exit** without saving changes.

IMPORTANT: Login sequences that don't have a method associated with them are not saved.

---

9 Click **OK**.

Deleting a Login Sequence

1 Launch OpenText Identity Console.

2 Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.

3 Go to **Authentication Management** > **Login Method Sequences**.

4 Select the **Login Sequence** you want to delete, then click **Delete**  to delete or remove the login method Sequence.

If one or more login sequences still use the login method, a warning message appears.

5 Click **OK**.

Authorizing Login Sequences for Users

- "Assigning Login Sequences" on page 613
- "Authorizing a Login Sequence" on page 614

Assigning Login Sequences

Authorized and default login sequences can be assigned to a user, a container, a partition root, or the login policy object. NMAS searches for the authorized or default login sequences for a user by attempting to read the attributes from first the User object, then the container of the user object, then the partition root of the user object, and finally the login policy object.

The attributes found with the User object supersede any attributes found with container, partition root, or login policy object. If a login sequence has been assigned to a partition root, that login sequence applies to all the users under that partition root only if a login sequence has not already been individually assigned to specific users.

Also, a login sequence assigned to a container applies only to the users with unassigned sequences in that container, and not to the users in subcontainers of that container.

Authorizing a Login Sequence

1. Launch OpenText Identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.
3. go to **Authentication Management** tile > **Login Method Sequences.**
4. Select **Login Sequence**, click Authorize Login Sequence Method  to authorize or click Unauthorize Login Sequence Method  to unauthorize the selected login method sequence. Or click down – arrow to authorize or unauthorize the sequence and click upside -down arrow to reorder the sequence list.

   Under **Default** list, only authorized Login Sequence Methods can be set to Default. You can use this toggle icon  to perform this action.

Setting Default Login Sequences

To set a default login sequence so that users are not required to specify a login sequence when logging in:

1. Launch OpenText Identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.
3. go to **Authentication Management** tile > **Login Method Sequences**.
4. Select an authorized login sequence under **Default** list, then click .

   The sequence you select will be the default login sequence. If a user attempts to log in without using a login sequence, this default login sequence is used.

NOTE: If a workstation is unable to execute the user's default login sequence, the NDS password login method is used.

For more information on how to assign login sequences, see "Assigning Login Sequences" on page 613.

Deleting a Login Method

The NMAS in OpenText Identity Console does not allow you to delete a login method if that method is part of any login sequence. The default installation of a login method creates a login sequence containing only that method. As a result, most methods exist in at least one sequence.

NOTE: nmasinst does not have an option to remove NMAS methods. It must be done through OpenText Identity Console.

To delete a login method, you must complete the following two procedures:

- "Removing the Login Method from Any Login Sequence" on page 615
- "Deleting the Login Method" on page 615

Removing the Login Method from Any Login Sequence

To use OpenText Identity Console to remove the login method for any login sequence:

1. On the OpenText Identity Console home page > click **Authentication Management** > **Login Method Sequences**.
2. For each sequence in the **Login Method Sequences** list:
   - **2a** Click the sequence name.
   - **2b** Verify that the login method you will be deleting is not listed in the **Login Methods** or **Post-Login Methods** lists.
   - **2c** If the login method is listed as one of the selected methods, you can move it from the list by selecting it and clicking the left-arrow.

When the login method has been removed from all login sequences, you can then delete it. See "Deleting the Login Method" on page 615.

Deleting the Login Method

To use OpenText Identity Console to delete the login method:

1. On the OpenText Identity Console home page > click **Authentication Management** > **Login Methods**.
2. Select the login method or methods you want to delete.
3. Click **Delete**, then click **OK**.

Deleting a Login Sequence

1. Launch OpenText Identity Console.
2. Authenticate to the OpenText eDirectory tree as an administrator or a user with administrative rights.
3. Go to **Authentication Management** > **Login Method Sequences**.
4. Select the **Login Sequence** you want to delete, then click **Delete**  to delete or remove the login method Sequence.

   If one or more login sequences still use the login method, a warning message appears.
5. Click **OK**.

Using NMAS to Log In to the Network

After NMAS is installed, you are ready for users to log in to the network. This section describes some of the additional features of the login experience that you should communicate to your network users.

- "Password Field" on page 616
- "Advanced Login" on page 616
- "Unlocking the Workstation" on page 617
- "Capturing an NMAS Client Trace" on page 617
- "Viewing NMAS Clearance Status" on page 617

Password Field

Depending upon how the NMAS client software was installed, there might or might not be a password field in the Novell Client login dialog box. If users are using a biometric or physical device () login factor, they might not need a password to log in to the network.

See the Novell Client For Windows documentation (http://www.novell.com/documentation/windows_client/) for more information on hiding the password field.

Advanced Login

Those using NMAS login methods to log in to the network can customize the login by selecting a desired clearance and login sequence. Otherwise, the last login sequence and clearance (if any) are used. If no clearance or login sequence has been previously specified, the defaults are used.

1. When the Novell Client dialog box appears, click **Advanced**.
2. Click the **NMAS** tab.
3. Select the desired login sequence from the **Login** drop-down list or browse the OpenText eDirectory tree for a complete and current list.

   You can browse only if an OpenText eDirectory tree has been specified on the **OpenTxt eDirectory** tab.
4. Specify the desired user session clearance or browse the OpenText eDirectory tree for a complete and current list.

   By default, the **Clearance** field is disabled. To enable the **Clearance** field:

   4a Right-click the red N in the task bar.

   4b Click **Novell Client Properties** > **Location Profiles**.

   4c Select the desired profile, click **Properties**, then click **Properties**.

   4d On the **NMAS** tab, select **Display Clearance Field**.

   4e Click **OK** three times.

   IMPORTANT: Users might have multiple session clearances for each login sequence. Make sure that the **Clearance** field is filled in with the desired user session clearance.
5. Click **OK**.

Unlocking the Workstation

With the addition of NMAS to a user's workstation, the process to unlock Windows workstations changes. Normally, users can enable password protection for their workstations by using a screen saver configured from the Windows Display control panel. To unlock a workstation with NMAS, users must instead go through the same authentication process used to originally log in.

For example, if you used NMAS to authenticate to the network and you used a biometric login method, you must use the same biometric login method again to unlock and use the workstation.

If you are using a Windows workstation, you must unlock the workstation using the login method that was used to log into the tree. If you have connections to multiple OpenText eDirectory trees, the login sequence for any OpenText eDirectory tree can be used. The default is the first OpenText eDirectory tree.

Capturing an NMAS Client Trace

Capturing an NMAS client trace can help in troubleshooting NMAS authentication problems. For more information, see TID # 3331372.

Viewing NMAS Clearance Status

1. Right-click the red N in the task bar.
2. Click **Novell Connections**.
3. Scroll over to view the NMAS clearance associated with each connection.

History of OpenText Passwords

In the past, administrators have had to manage multiple passwords (simple password, NDS password, enhanced password) because of password limitations. Administrators have also needed to deal with keeping the passwords synchronized.

- NDS Password: The older NDS password is stored in a hash form that is non-reversible. Only the NDS system can make use of this password, and it cannot be converted into any other form for use by any other system.
- Simple Password: The simple password was originally implemented to allow administrators to import users and passwords (clear text and hashed) from foreign LDAP directories such as Active Directory* and iPlanet*.

  The limitations of the simple password are that no password policy (minimum length, expiration, etc.) is enforced.
- Enhanced Password: The enhanced password (no longer supported), the forerunner of Universal Password, offers some password policies, but its design is not consistent with other passwords. It provides a one-way synchronization and it replaces the simple or NDS password.

Universal Password was created to address these password problems. It provides:

- One password for all access to OpenText eDirectory.
- Enables the use of extended characters in password.

- Enables advanced password policy enforcement.
- Allows synchronization of passwords from OpenText eDirectory to other systems.

Universal Password is managed by the Secure Password Manager, a component of the NMAS module. Secure Password Manager simplifies the management of password-based authentication schemes across a wide variety of OpenText, Novell, and OpenText partner products. The management tools only expose one password and do not expose all of the behind-the-scenes processing for backwards compatibility.

Secure Password Manager and the other components that manage or make use of Universal Password are installed as part of the OpenText eDirectory install. However, Universal Password is not enabled by default. Because all APIs for authentication and setting passwords are moving to support Universal Password, all the existing management tools, when run on clients with these new libraries, automatically work with the Universal Password.

NOTE: The Password Policies tile is available under Authentication Management.

The Novell Client supports the Universal Password. It also continues to support the NDS password for older systems in the network. The Novell Client has the capability of automatically migrating the NDS password to the Universal Password at the time of the first login.

The password expiration time is not updated when the NDS password is migrated to the Universal Password unless the “Verify whether existing passwords comply with the password policy (verification occurs on login)” password policy rule is set to “true”.

For more information about deploying and managing Universal Password, see Chapter 26, “Managing Passwords,” on page 707.

NMAS HOTP Based Login

The following sections contain information about the NMAS HOTP:

- “Overview” on page 618
- “Installation” on page 619
- “Resynchronization of the Counter” on page 621
- “Configuration” on page 622
- “Known issues” on page 623
- “nmashotpconf utility cannot modify the user resynchronization window” on page 623

Overview

HOTP is an HMAC-based one-time password (OTP) algorithm. An OTP is a password that is valid for only one login session or transaction. An OTP provides better performance than the traditional (static) passwords because there are less chances of security attacks associated with it. A potential intruder who records an OTP that has been used to log into a service or to conduct a transaction, cannot manipulate it because it has already been used once and is no longer valid.

Every OTP based authentication requires an OTP server and an OTP client (hardware/software). Implementation of OTP based authentication in NMAS is based on the RFC 4226 standard. Traditionally, the NDS password that was individually presented to the server is now appended to the OTP to enhance the password based authentication by retaining all the client components and their user interface.

The authentication to OpenText eDirectory server is done through the HOTP feature by using LDAP-based login.

LDAP-Based Login

Prerequisites

- ensure that the `NDSD_TRY_NMASLOGIN_FIRST` environment variable to set to true.

  For more information, refer to the Managing Passwords.

NOTE: This is set by default with OpenText eDirectory 9.0 or later.

Login Method

An HOTP-enabled user can perform LDAP bind by concatenating the NDS password with the HOTP value.

For example,

```
ldapsearch -D cn=user1,o=novell -w secret40338314 -h 164.99.91.165 -p 389 -
b   "o=novell" -s sub -LLL dn
```

NCP-Based Login

A HOTP-ready/enabled user can perform NCP login by concatenating the NDS Password with the HOTP value by using any of the following utilities:

- ndslogin

  For example,

  ```
  ndslogin user1.org -h org.com -p secret40338314
  ```

- OpenText Identity Console
- iMonitor

NOTE: OpenText Identity Console that perform LDAP authentication will fail if used by HOTP-enabled users.

Installation

- “Server Installation” on page 620
- “Obtaining and Using nmashotpconf Utility” on page 620

Server Installation

The HOTP server module is a part of the NMAS server component. The server module validates the OTP presented from the client.

The following attributes are available on the NMAS HOTP server:

- `sasOTPCounter` (per user attribute)
- `sasOTPEnabled` (per user/immediate parent container/partition root/Login Policy object)
- `sasOTPDigits` (per user/immediate parent container/partition root/Login Policy object)
- `asOTPLookAheadWindow` (tree wide set at the Login Policy object)
- `sasOTPResync` (9 per user attribute)

Obtaining and Using nmashotpconf Utility

The `nmashotpconf` utility is a configuration utility that configures the OTP attributes on the eDirectory server.

NOTE: The HOTP utility is available only for the Linux 64-bit platforms.

To execute the `nmashotpconf` utility, perform the following steps:

1. Obtain the `nmashotpconf` utility and specify the directory where you unzipped the NMAS HOTP utility.

   NOTE: `nnmashotpconf` utility is bundled with the NMAS. To download this utility, refer https://sld.microfocus.com.

   The unzipped file contains the `linux` and `linux_x64` directories for the 32-bit and 64-bit Linux machines.

   The `linux` and `linux_x64` directories contain the `nmashotpconf` executable and `libnmasext.so` files.

2. Go to the `linux/final` directory on a Linux 32-bit machine, else go to the `linux_x64/final` directory on a Linux 64-bit machine.
3. Download the trusted root certificate and store it locally.

   For more information, see “Exporting a Trusted Root or Public Key Certificate” on page 663.

   For usage,

```
 nmashotpconf -h <host_name> [-p <ssl_port>] -D <login_dn> [-w
<password>]
  -e <trusted_cert> -t <cert_type> [-r <resync_window>] [-y
  <user_resync_window>] [-u <hotp_dn> [-o <hotp_options>] [-d digits]
[-c
  <counter>] [-s <secret> -f <secret_format>]]
```

Option	Description
host_name	Specifies the LDAP server name or the IP address of the server.

Option	Description
ssl_port	Specifies the SSL port on the LDAP server. The default value is 636.
login_dn	Specifies the DN for the user.
password	Specifies the password for the user DN.
trusted_cert	Specifies the trusted root certificate file.
cert_type	Specifies the trusted root certificate encoding type. For example, DER means der-encoded file, and B64 means b64-encoded file.
encoded file digits	Specifies the number of digits used as the HOTP value. **NOTE:** This setting is applicable to all the users in the tree.
resync_window	Specifies the counter re-synchronization look-ahead window.
user_resync_window	Specifies the counter user re-synchronization look-ahead window.
hotp_dn	Specifies the target DN for which you are configuring the HOTP attributes. To configure the HOTP at the tree level, enable/disable HOTP at the tree level, or configure digits at tree level, then specify the DN as `cn=Login Policy,cn=Security`.
hotp_options	Enables or disables the HOTP for the hotp_dn option. Specify ENABLE to enable the HOTP, and DISABLE to disable HOTP.
counter	Specifies the HOTP counter value. The vaild range of the counter value is between 0 and 2147483647. The counter value is set through the hotp_dn option.
hotp_dn secret	Specifies the OATH HOTP secret. For example, the raw byte value of secret in the hexadecimal format is 31323334353637383930313233343536373839 30, or the corresponding ASCII/Extended ASCII string is 12345678901234567890.
secret_format	Specifies the format of the OATH HOTP secret. ◆ **STRING:** This format is used for an ASCII/Extended ASCII string. For example, 12345678901234567890. ◆ **RAW:** This format is used for raw byte values in a hexadecimal format. For example, 31323334353637383930313233343536373839 30, where hexadecimal value of the first character is 31, the value of the second character is 32, and so on.

Resynchronization of the Counter

The counter value of the server is incremented only after successful HOTP authentication, and the counter on the client is incremented every time a new HOTP is requested by the user. The counter values on the server and the counter on the client might be out of synchronization.

To address this, you should have a tree-wide look-ahead or a resynchronization window setting in place. If the server finds that the received HOTP does not correspond to the server counter value, the server can recalculate the next few HOTP values that are within the resynchronization window, and check them against the received HOTP. If there is a match, authentication succeeds and the server counter is set to the counter value that corresponds to the matched HOTP.

For successful authentication the server counter is set to the next counter value at which the authentication succeeds.

The tree-wide resynchronization window setting should be as low as possible in order to restrict the space of possible solutions for an attacker trying to recreate the HOTP values.

If the mismatch between the client and server counters is beyond the tree-wide resynchronization window setting, resynchronization can be achieved by temporarily setting a user-specific resynchronization window to a large value and then attempting an HOTP-based authentication.

The nmashotpconf utility should be used for configuring HOTP-based authentication. For more information, read the Configuration section.

Configuration

To provision an OpenText eDirectory user for an HOTP-based authentication, do the following configuration settings according to the RFC 4226 standard.

- Enable HOTP on the user/container/partition root/Login Policy object in the same order of precedence.
- Set the HOTP-shared secret key and counter on the user. These two settings together determine the HOTP value.
- Configure the number of digits in HOTP values on the user/ container/partition root/Login Policy object.The valid range of digits is from 6 to 9.
- Set the resynchronization windows as follows:
  - Set the tree-wide resynchronization window at the Login Policy object.
  - Set the user-specific resynchronization window at the user level. This is needed only when the client and server are out of sync.

Examples:

- To configure a secret and a counter on the user object, run the following command:

```
./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell   -
e   /var/opt/novell/eDirectory/data/SSCert.der  -t  DER  -u
cn=user1,o=novell  -c 0   -s   31323334353637383930313233343536373839 30
-f  RAW
```

- To enable the OTP for a user object, run the following command:

```
./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell   -
e   /var/opt/novell/eDirectory/data/SSCert.der  -t  DER  -u
cn=user1,o=novell  -o ENABLE
```

- To disable the OTP for a user object, run the following command:

```
./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell   -
e   /var/opt/novell/eDirectory/data/SSCert.der  -t  DER  -u
cn=user1,o=novell  -o DISABLE
```

Similarly, you can enable or disable the OTP for a container/partition or a root/Login Policy object.

- To configure an OTP digit for a user object, run the following command:

  ```
  ./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell   -
  e /var/opt/novell/eDirectory/data/SSCert.der  -t  DER  -u
  cn=user1,o=novell  -d 6
  ```

  Similarly, you can set the OTP digit for a parent container/partition root/ Login Policy object.

- To configure the user resyncronization window, run the following command:

  ```
  ./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell   -
  y 5 -e   /var/opt/novell/eDirectory/data/SSCert.der  -t  DER  -u
  cn=user1,o=novell
  ```

- To configure the counter re-synchronization look ahead window, run the following command:

  ```
  ./nmashotpconf -h 192.168.1.1 -p 636 -D cn=admin,o=novell -w novell
  -r 6
  ```

NOTE: To test the configuration, you can use HOTP s generated by any hardware or software which is compliant to the HOTP standards.

Known issues

- "ndsconfig add fails for an HOTP enabled administrative user" on page 623
- "Login through HOTP-enabled user to a read-only replica fails" on page 623

ndsconfig add fails for an HOTP enabled administrative user

For HOTP enabled users, the OTP digit is used for authentication. The ndsconfig utility uses the same OTP digit for subsequent authentication, which causes the ndsconfig add to fail. Similarly, ndsconfig upgrade also fails.

To work around this issue, do not enable HOTP for the user through which you are performing ndsconfig add/ upgrade.

Login through HOTP-enabled user to a read-only replica fails

If you perform LDAP login through the HOTP-enabled user by sending a request to the read-only replica, the LDAP chaining does not happen. The read-only replica does not forward the request to the server where the actual user resides. The replica fails giving an illegal replica type error.

nmashotpconf utility cannot modify the user resynchronization window

If the value of the user resynchronization window is already set (say 2) and its value is changed by using the nmashotpconf utility, it displays the following error:

```
ldap_modify_ext_s on HOTP DN failed: error code=19: Constraint violation
```

One of the reasons for the error could be using a combination of the **-o** (the OTP enable or disable option), **-d** (OTP digit), **-c** (otpcouter) and **-y** (user_resync_window) options for modifying the user resynchronization value.

Other Administrative Tasks

This section describes other administrative tasks for NMAS:

- "Using the Policy Refresh Rate Command" on page 624
- "Using the LoginInfo Command" on page 624
- "Disabling the NMAS Based Logins for LDAP" on page 628
- "Invoking NMAS Commands" on page 628
- "Setting the Delay Time for Failed Login Attempts" on page 628
- "Using DSTrace" on page 629
- "Disabling and Uninstalling the NMAS Client" on page 629
- "Using External Certificates with OpenText Audit" on page 629

Using the Policy Refresh Rate Command

You can configure NMAS to refresh the cached NMAS login policy from the NMAS login policy stored in the Security container at scheduled intervals instead of upon every login attempt. This configuration is set per server by using the NMAS policy refresh rate command.

NOTE: The server accesses the Security container once during startup to cache the policy. Then, based on the configured intervals, the server attempts to access the Security container to refresh the policy.

The policy refresh rate command has the following syntax:

```
nmas RefreshRate minutes
```

where *minutes* is the number of minutes between each attempt to check if the cached NMAS login policy needs to be updated.

For information on how the policy refresh rate command can be invoked for each NMAS Server platform, see "Invoking NMAS Commands" on page 628.

Using the LoginInfo Command

With NMAS 3.2 or later, you can turn off automatic updating of certain user object login attributes by using the `LoginInfo <numb>` command. You might want to do this manually if automatically updating attributes causes problems. The following sections further explain this functionality:

- "NMAS Login for LDAP Bind" on page 625
- "Problems Caused by Automatically Updating User Object Login Attributes" on page 625

- "Using the LoginInfo Command to Control LoginInfo Attributes When Attributes are Updated" on page 625
- "Using the sasUpdateLoginInfo and sasUpdateLoginTimeInterval Attribute" on page 626

NMAS Login for LDAP Bind

NMAS login is enabled for LDAP Bind by default with OpenText eDirectory 9.2. When NMAS login is enabled, OpenText eDirectory automatically updates user object login attributes after the user has authenticated. The following is a non-exhaustive list of login attributes that are updated:

- Login Time
- Network Address
- Last Login Time

To disable NMAS based login for LDAP, refer "Disabling the NMAS Based Logins for LDAP" on page 628.

Problems Caused by Automatically Updating User Object Login Attributes

The automatic updating of user object login attributes can lead to the following problems:

- High utilization
- Unresponsiveness
- Client time-outs seen on busy authentication servers, especially in LDAP environments

If you are experiencing these problems, you might want to regulate when the login attributes are updated. For information on how to do this, see "Using the LoginInfo Command to Control LoginInfo Attributes When Attributes are Updated" on page 625.

Using the LoginInfo Command to Control LoginInfo Attributes When Attributes are Updated

To control when login attributes are updated, execute the `nmas LoginInfo <num>` command.

The value for <num> is as follows:

- **0 or off:** Do not update any login attributes.
- **1:** Only update attributes that are required by intruder detection.
- **2:** Update all login attributes except unused user password policy attributes.
- **3 or on:** Update all login attributes.

For information on how to invoke the LoginInfo command for each NMAS Server platform, see "Invoking NMAS Commands" on page 628.

Using the sasUpdateLoginInfo and sasUpdateLoginTimeInterval Attribute

The `sasUpdateLoginInfo` attribute controls the updates of `LoginInfo` attributes.

The `sasUpdateLoginTimeInterval` attribute controls the update of the `Login Time` attribute of a user for a specified interval.

The `sasUpdateLoginInfo` attribute can have the following values:

- **0 or off:** Do not update any login attributes.
- **1:** Only update attributes that are required by intruder detection.
- **2:** Update all login attributes except unused user password policy attributes.
- **3 or on:** Update all login attributes.

The `sasUpdateLoginTimeInterval` attribute can have values from 0 to 1440 minutes (that is, one day).

- If the value is 0, the `Login Time` and `Last Login Time` attributes are updated for every successful login.
- If the value is between **1** and **1440** minutes, the `Login Time` attribute is updated after the specified interval. The Last Login Time attribute will not be updated.

NOTE: The `Login Time` attribute is not updated on consecutive successful logins during the interval. However, if there is a login failure during the interval followed by successful login, the `Login Time` attribute will be updated. The interval time from the successful login is counted.

The `sasUpdateLoginTimeInterval` attribute is effective only if the `sasUpdateLoginInfo` attribute value is set to 2 or 3.

The attributes can be specified for the following objects in the order of precedence (user having the highest precedence).

- User
- Container of the user
- Partition root
- Login Policy

If the `sasUpdateLoginInfo` and `sasUpdateLoginTimeInterval` are set on the Login Policy object, the setting becomes effective after the next policy refresh cycle. If the attributes are not set for the user, container, partition root, or Login Policy, the value set on a server using command line is used to maintain backward compatibility.

Following is an example to set the attribute values on the eDirectory server:

```
#cat nmas.config (The nmas.config file must be in the same directory as the
dib directory.)
nmas LoginInfo 2
nmas UpdateLoginTimeInterval 30
```

To set attributes value at the partition root:

1. To add the attributes to the Tree, go to **Identity Console** > **Schema Management** tile > **Attribute** > click **Create Attribute**.
2. Use the arrow to move the required attribute from **Available optional attribute** list to **Optional attribute** list.

To set the values of the attribute at partition root, run the `ldapmodify` command and the following commands at the command line or using an `ldif` file:

```
dn:T=< tree name>
changetype:modify
add:sasUpdateLoginTimeInterval
sasUpdateLoginTimeInterval:35

dn:T=< tree name>
changetype:modify
add:sasUpdateLoginInfo
sasUpdateLoginInfo: 2
```

You can edit the `sasUpdateLoginInfo` or `sasUpdateLoginTimeInterval` attribute values for user, container, and Login Policy objects using OpenText Identity Console or an `ldif` file.

Example:

```
#cat changesasUpdateLoginInfo.ldif
dn: cn=user1,o=org
change type: modify
replace: sasUpdateLoginInfo
sasUpdateLoginInfo: 1

#cat changesasUpdateLoginTimeInterval.ldif
dn: cn=user1,o=org
changetype: modify
replace: sasUpdateLoginTimeInterval
sasUpdateLoginTimeInterval: 60
```

The setting disables the update of `Login Time` attribute of user1 for 60 minutes from the previous update of the attribute.

To specify the `sasUpdateLoginInfo` and `sasUpdateLoginTimeInterval` attributes from OpenText Identity Console:

1. In OpenText Identity Console, click the **Object Management** tile.
2. From the **Type** drop down list select User > click **Search**.
3. Select a user.
4. On the **Others** tab, click **Add Other Attributes** +, select **sasUpdateLoginTimeInterval** from **Unvalued Attributes** list.
5. Use the arrow button to move **sasUpdateLoginTimeInterval** from **Unvalued Attributes** list to the **Valued Attributes** list, then click **Apply**.

Disabling the NMAS Based Logins for LDAP

The NMAS login is enabled by default in OpenText eDirectory 9.2. To disable the NMAS login, set `NDSD_TRY_NMASLOGIN_FIRST` to `false`.

To disable NMAS based login for LDAP on Windows, Right-click My Computer and select Properties. In the Advanced tab click Environment Variables. Under System Variables, add the variable and set the value to false.

NOTE: You must add all the environment variables required for the OpenText eDirectory service in the env file located in the `/etc/opt/novell/eDirectory/conf` directory on RHEL 7.x and SLES 12.x platforms.

Invoking NMAS Commands

How you invoke an NMAS command differs depending on what platform you are running. The following platforms are supported:

- "Windows" on page 628
- "Linux" on page 628

Windows

When NMAS is started, it processes the commands in the `nmas.cfg` file. The `nmas.cfg` file must be in the same directory as the `dib` files, which are usually in `c:/novell/nds/dibfiles`.

or

After NMAS has been started, use the following procedure:

1. In the OpenText eDirectory Services console, select `nmas.dlm`.
2. Type the command in the **Startup Parameters** field.
3. Click **Configure**.

Linux

When NMAS is started, it processes the commands in the `nmas.config` file. The `nmas.config` file must be in the same directory as the `dib` directory. For example, if the `.dib` directory path is `/var/opt/novell/eDirectory/data/dib`, then the `nmas.config` file path is `/var/opt/novell/eDirectory/data/nmas.config`.

Setting the Delay Time for Failed Login Attempts

1. On the OpenText Identity Console home page > click **Objects Management** tile.
2. Click **Type** drop down menu > select **User** > On the **Modify User** page > click **Authentication** tab > **Settings** drop down.
3. Type the number of seconds you want the login screen to be delayed between failed login attempts, then click **Save**.

Using DSTrace

You can use the DSTrace utility to get trace information from NMAS.

For information on how to capture an NMAS client trace, see TID # 3331372.

For information on how to capture an NMAS server trace, see TID # 3815371.

Disabling and Uninstalling the NMAS Client

To disable the NMAS Client:

1. On the workstation, right-click the Red N.
2. Click **Novell Client Properties**.
3. Click the **Advanced Login** tab.
4. From the **Parameter Groups** list, select **NMAS Authentication**.
5. Under **Setting**, select **Off**.
6. Click **OK**.

To uninstall the NMAS Client, use the Add/Remove Programs option of the Windows Control Panel.

NOTE: Disabling or removing NMAS does not remove support for changing the Universal Password from the Novell Client for Windows.

Using External Certificates with OpenText Audit

To use an external certificate with NMAS and OpenText Audit, you must first convert the certificate into two `.pem` files with the following names:

- `nmascert.pem`: This is the file containing the certificate.
- `nmaskey.pem`: This is the file containing the private key.

These files need to be copied to the following directories on each platform for each NMAS server in the system:

- Linux: `/etc`
- Windows: the return from GetWindowsDirectory (typically `c:\windows`)

NMAS provides the `nmascert.pem` and the `nmaskey.pem` files to the OpenText Audit platform agent when the log is open, if they exist. If the files don't exist, NMAS provides the internal certificate and key to the OpenText Audit platform agent.

Security Considerations

This section contains specific information related to security with NetIQ Modular Authentication Services. It contains the following subsections:

- "Partner Login Methods" on page 630
- "Login Policies" on page 630

- "NMASInst" on page 631
- "Universal Password" on page 631
- "SDI Key" on page 632

Partner Login Methods

OpenText has not evaluated the security methodologies of partner login methods. Although the partner products might have qualified for the OpenText Yes, Tested & Approved or OpenText Directory Enabled logos, those logos relate to general product interoperability only.

Login Policies

- If authorized login sequences, default login sequences, authorized clearances, or default clearances are assigned to a container that is not a partition root, the policy is only effective for user objects in the container, and not for user objects in subcontainers.
- If authorized login sequences, default login sequences, authorized clearances, or default clearances are assigned to a container that is a partition root, the policy is effective for all users in the partition that do not have these values assigned to the user object or to the object's parent container.
- If authorized login sequences, default login sequences, authorized clearances, or default clearances are assigned to a Login policy, that policy is effective for all users in the tree that do not have these values assigned to the user object, to the object's parent container, or to the object's partition root.
- When users are assigned passwords or other guessable login secrets such as challenge question responses, you should enable intruder detection to slow down or prevent intruders from guessing the login secrets.
- By default, failed login attempts are delayed by three seconds. This delay is intended to slow down the attempts of intruders to guess passwords. The length of the failed login delay is configurable. You should use the default of three seconds.
- Login policies such as intruder detection, network address restrictions, and time of day restrictions are enforced for all login sequences. For example, the login policies are enforced when the forgotten password self-service feature of several OpenText products invokes the challenge/response login method.
- You should enable NMAS™ Auditing so that you can track login attempts and changes in configuration.
- Using the policy refresh rate command to check if the cached password policy needs to be refreshed on defined intervals instead of during each login causes a delay in the application of login policy changes.
- The `LoginInfo` command can be used to disable updating login-related attributes during login. These attributes include the intruder detection attributes. Disabling the update of these login-related attributes improves login performance. However, disabling the update of these attributes might lessen the security of the system.
- The intruder detection policy can be set on the user object's direct container or on the user object's partition root. NMAS checks the parent container first for an intruder detection policy. If no policy is found, then the partition root is checked for an intruder detection policy.

NMASInst

When you are upgrading a login method, nmasinst replaces a newer version with the older version unless the `-checkversion` option is used.

Although nmasinst provides an option to specify the password on the command line, it is not recommended because the password could be compromised. With OpenText eDirectory 9.0 or later, nmasinst allows you to retrieve a password from either file or an environment variable.

Universal Password

- Because the Security container contains global policies, you should be careful where you place writable replicas. Some servers can modify the overall security policies specified in the OpenText eDirectory tree. In order for users to log in with NMAS, replicas of the User objects and security container must be on the NMAS server.
- If a Password policy is assigned to a container that is not a partition root, that policy is only effective for the user objects in the container, and not for user objects in subcontainers.
- If a Password policy is assigned to a container that is a partition root, that policy is effective for all users in the partition that do not have these values assigned to the user object or to the object's parent container.
- If a Password policy is assigned to a Login policy, that policy is effective for all users in the tree that do not have these values assigned to the user object, to the object's parent container, or to the object's partition root.
- The password expiration time is not updated when the NDS password is migrated to the Universal Password unless the “Verify whether existing passwords comply with the password policy (verification occurs on login)” password policy rule is set to “true”.
- Password policies can be configured to allow the user or a password administrator to read the Universal Password by using documented NMAS LDAP extensions. These options should not be enabled unless required for your specific installation. If you require user passwords to be readable, you should configure the Password policy to only allow selected users to read the passwords.
- You should configure a password policy to synchronize to the Distribution Password only if Identity Manager Password Synchronization is being used to synchronize passwords between connected systems.

  For more information on sychronizing passwords between connected systems using Identity Manager Password Synchronization, see the *OpenText Identity Manager 4.5 Password Management Guide*.
- You should only configure a password policy to synchronize to the Simple Password only if:
  - You have servers that hold a writable replica of user objects
  - Users access those servers using Native File Access Protocols such as CIFS and AFP.
- When advanced password rules are enabled for a password policy, the legacy password rules on the User object are ignored, and are updated to match the password policy rules when users change their passwords or log in.
- The password exclusion rules (password history, excluded passwords, and disallowed attribute vales) are not enforced when NMAS is used to generate random passwords.

- When selecting password rules, you should balance the requirements for hard-to-guess passwords with hard-to-remember passwords.
- When an administrator specifies that the NDS Password is to be removed, the result is that the NDS Password Hash is set to a random value that is unknown to anyone but OpenText eDirectory. There might or might not be a password value that could be hashed to that random value.
- XML Password Complexity
  - If there are duplicate rule tags, the most restrictive rule is used (others are ignored) for checking passwords against the policy and for random password generation.
  - The `ViolationsAllowed` and `NumberOfCharactersToEvaluate` rule set attributes are ignored for random password generation.
  - Only the first policy in an XML policy is used for random password generation.

For additional information on Universal Password security, see Chapter 26, “Managing Passwords,” on page 707.

SDI Key

You should make the Security Domain Infrastructure (SDI) key, also known as the tree key, a Triple DES key (3DES). The SDI key can be checked and upgraded by using the SDIDiag utility. See Verify that the SDI Domain Key Servers are running NICI *3.0.* in the Chapter 26, “Managing Passwords,” on page 707.

OpenText eDirectory 9.0 onwards, AES 256 tree key is also supported. For more information, see Creating an AES 256-Bit Tree Key.

25 Understanding the Certificate Server

OpenText Certificate Server is automatically installed when you install OpenText eDirectory. Certificate Server provides public key cryptography services that are natively integrated into OpenText eDirectory and that allow you to mint, issue, and manage both user and server certificates. These services allow you to protect confidential data transmissions over public communications channels such as the Internet.

NOTE: ◆If you are unfamiliar with public key cryptography concepts, see "Public Key Cryptography Basics" on page 695.

- MD2 and MD5 signature algorithms for RSA encryption are not supported with OpenText eDirectory 9.2 and above.

- "OpenText Certificate Server Features" on page 633
- "OpenText Certificate Server Components" on page 634
- "Setting Up OpenText Certificate Server" on page 641
- "Managing OpenText Certificate Server" on page 650
- "Public Key Cryptography Basics" on page 695
- "Entry Rights Needed to Perform Tasks" on page 702

OpenText Certificate Server Features

Public key cryptography presents unique challenges to network administrators. OpenText Certificate Server helps you meet these challenges in the following ways:

- Provides public key cryptography services on your network

  You can create an Organizational Certificate Authority (CA) within your OpenText eDirectory tree, allowing you to issue an unlimited number of user and server certificates. You can also use the services of an external certificate authority, or use a combination of both as your needs dictate.

- Controls the costs associated with obtaining and managing public key certificates

  You can create an Organizational CA and issue public key certificates through the Organizational CA.

- Allows public key certificates to be openly available while also protecting them against tampering

  Certificates are stored in OpenText eDirectory and can therefore leverage eDirectory replication and access control features.

- Allows private keys to be accessible to only the software routines that use them for signing and decrypting operations

  Private keys are encrypted by NICI and made available only to the software routines using them for signing and decrypting operations.

- Securely backs up private keys.

  Private keys are encrypted by NICI, stored in OpenText eDirectory, and backed up by using standard OpenText eDirectory backup utilities.
- Allows central administration of certificates using OpenText Identity Console.

  OpenText Identity Console allows you to manage certificates issued from your Organizational CA or from any other CA that supports a certificate signing request in PKCS #10 format.
- Allows users to manage their own certificates

  Users can use Identity Console to export keys for use in cryptography-enabled applications without system administrator intervention.
- Supports popular e-mail clients and browsers

OpenText Certificate Server Components

This section describes the components of OpenText Certificate Server.

- "OpenText Certificate Server" on page 634
- "NICI" on page 640

OpenText Certificate Server

OpenText Certificate Server consists of the PKI server component in OpenText Identity Console that is administration interface for Certificate Server.

Certificate Server allows you to do the following tasks:

- Establish an Organizational Certificate Authority that is specific to your OpenText eDirectory tree and your organization.
- Request, manage, and store public key certificates and their associated private keys in the OpenText eDirectory tree.

Using 8192 Bit RSA Keys in Certificates

Using the Certificate Server, you can select the key size as part of any certificate creation procedure. OpenText eDirectory supports key sizes up to 8192 bits. If you plan to use X.509 certificates with a 8192 bit RSA public key for your applications, the applications must support 8192 bit RSA keys. Otherwise, your applications may not function as expected.

IMPORTANT: Using an X.509 certificate with 8K bit RSA public keys for establishing TLS connection impacts the performance of your OpenText eDirectory servers. OpenText does not recommend configuring OpenText eDirectory servers to use RSA certificates with 8K bit keys because TLS session establishment can be computation intensive for the servers and it may cause significant system slowdown when TLS sessions are established concurrently.

Ensure that you upgrade all the servers in your eDirectory tree to 9.1 before creating a CA certificate with a 8192 bits RSA public key.

NOTE: Ensure that you are using OpenText eDirectory 9.1, Identity Console 1.7 at a minimum before configuring a server certificate with 8192 bit RSA public key.

Using ECDSA Certificates

The Certificate Server supports the use and management of Elliptic Curve Digital Signature Algorithm (ECDSA) certificates and keys in the same way as it supports the RSA certificates.

An ECDSA key pair whose security is comparable to an RSA key is significantly smaller than the RSA key and significantly improves the performance when used in establishing TLS connections. ECDSA makes use of elliptic curve cryptography (ECC). ECC generates keys through elliptic curves. The technology can be used in conjunction with most public key encryption methods, such as RSA and Diffie-Hellman. Using ECC-based signatures with digital certificates provide added size and performance advantages.

OpenText eDirectory supports ECDSA certificates with keys with the following curves:

- P-256
- P-384
- P-521

In Suite B mode, the Certificate Server adheres to RFC 5759. This RFC specifies that every Suite B certificate and CRL must be signed using ECDSA with keys generated using either P-256 or P-384 curves.

The signing CA's key must be on P-256 or P-384 curve if the certificate contains a key on P-256 curve. If the certificate contains a key on P-384 curve, the signing CA's key must be on P-384 curve. Any certificate and CRL must be hashed using SHA-256 or SHA-384, matched to the size of the signing CA's key.

After installing OpenText Certificate Server, you manage it by using OpenText Identity Console.

You can use OpenText Identity Console to perform the following tasks:

- “Create an Organizational Certificate Authority for Your Organization” on page 636
- “Create a Server Certificate Object for Each Cryptography-Enabled Application” on page 636
- “Create a User Certificate” on page 637
- “Create a Trusted Root Container” on page 637
- “Create a Trusted Root Object” on page 638
- “Create Certificates For External Users and Servers” on page 638
- “Validate Certificates” on page 638
- “Manage Certificate Revocation Lists” on page 638
- “Export Private Keys and Certificates” on page 639
- “Import Private Keys and Certificates” on page 640
- “Create an SAS Service Object” on page 640

Create an Organizational Certificate Authority for Your Organization

During the installation, you can elect to create an Organizational Certificate Authority (CA) if one does not already exist in the OpenText eDirectory tree. You can also create or re-create the Organizational CA after the installation is completed.

The Organizational CA object contains the public key, private key, certificate, certificate chain, and other configuration information for the Organizational CA. The Organizational CA object resides in the Security container in OpenText eDirectory.

After a server is configured to provide the certificate authority service, it performs that service for the entire OpenText eDirectory tree. If the tree has a subordinate CA certificate, and if you upgrade the server hosting subCA to OpenText eDirectory 9.2, ECDSA CA certificates are not generated. Administrator has to import a subordinate CA ECDSA certificate with the same subject name as the subordinate CA RSA certificate. If the server acting as a CA is on OpenText eDirectory 9.2, OpenText eDirectory creates the ECDSA certificates for the Organizational CA. eDirectory automatically creates ECDSA certificates for servers if the Organization CA has a ECDSA certificate.

For more information on creating an Organizational CA, see "Creating an Organizational Certificate Authority Object" on page 653.

Create a Server Certificate Object for Each Cryptography-Enabled Application

The Certificate Server installation creates default Server Certificate objects.

- SSL CertificateDNS - *server_name*
- A certificate for each IP address configured on the server (IP AG *xxx.xxx.xxx.xxx* - *server_name*)
- A certificate for each DNS name configured on the server (DNS AG *www.example.com* - *server_name*)
- SSL EC CertificateDNS - *server_name*
- A certificate for each IP address configured on the server (IP EC AG *xxx.xxx.xxx.xxx* - *server_name*)
- A certificate for each DNS name configured on the server (DNS EC AG *www.example.com* - *server_name*)

NOTE: OpenText eDirectory does not automatically create SSL CertificateIP. SSL CertificateDNS contains all the IPs listed in the Subject Alternative Name.

You can create other Server Certificate objects after the installation is completed.

The Server Certificate object contains the public key, private key, certificate, and certificate chain that enables SSL security services for server applications. Server Certificate objects can be signed by either the Organizational CA or by an external CA.

A server can have many Server Certificate objects associated with it. Any cryptography-enabled applications running on a particular server can be configured to use any one of the Server Certificate objects for that server. Multiple applications running on a given server can use the same Server Certificate object; however, a Server Certificate object cannot be shared between servers.

You can create Server Certificate objects only in the container where the server resides. If the Server object is moved, all Server Certificate objects belonging to that server must be moved as well. You should not rename a Server Certificate object. You can determine which Server Certificate objects belong to a server by searching for the server's name in the Server Certificate Object Name or by looking at the host server field when viewing the Server Certificate object in OpenText Identity Console.

The key pair stored in the Server Certificate object is referenced by the name you enter when the key pair is created. The key pair name is not the name of the Server Certificate object. When configuring cryptography-enabled applications to use key pairs, you reference those keys by their key pair name, not by the Server Certificate object name.

If the default Server Certificate objects become corrupted or invalid, use the Create Default Certificates Wizard to replace the old default certificates. For information on how to access the Create Default Certificates Wizard, see “Creating Default Server Certificate Objects” on page 661.

By default, OpenText eDirectory creates ECDSA certificates if the Organization CA has an ECDSA certificate.

Create a User Certificate

Users have access to their own user certificates and private keys, which can be used for authentication, data encryption/decryption, digital signing, and secure e-mail. One of the most common uses is sending and receiving digitally signed and encrypted e-mail using the S/MIME standard.

Generally, only the CA administrator has sufficient rights to create user certificates. However, only the user has rights to export or download the private key from OpenText eDirectory. Any user can export any other user's public key certificate.

The user certificate is created from the **Security** tab of the user's property page and is signed by the Organizational CA. Certificates and private keys created by other CAs can be imported after being created.

Multiple certificates can be stored on the user's object.

For more information on creating a user certificate, see “Creating a User Certificate” on page 649.

Create a Trusted Root Container

A trusted root provides the basis for trust in public key cryptography. Trusted roots are used to validate certificates signed by other CAs. Trusted roots enable security for SSL, secure e-mail, and certificate-based authentication.

A Trusted Root Container is an OpenText eDirectory object that contains Trusted Root objects.

The default Trusted Root Container is CN=trusted roots.CN=security.

For more information on creating a Trusted Root Container, see “Creating a Trusted Root Container” on page 649.

Create a Trusted Root Object

A Trusted Root object is an OpenText eDirectory object that contains a CA's Trusted Root certificate that is known to be authentic and valid. The Trusted Root Certificate can be exported and used as needed. Applications that are configured to use the Trusted Root Certificate consider a certificate valid if it has been signed by one of the CAs in the Trusted Root Container.

The Trusted Root object must reside in a Trusted Root Container.

For more information on creating a Trusted Root object see “Creating a Trusted Root Object” on page 650.

Create Certificates For External Users and Servers

The CA administrator can use the Organizational CA to sign certificates for users and servers outside of OpenText eDirectory. Such certificates are requested using a PKCS#10 Certificate Signing Request (CSR) provided to the CA administrator in an out-of-band fashion.

Given a CSR, the CA administrator can issue the certificate by using the Issue Certificate tool in OpenText Identity Console. The resulting certificate is not stored in an object in OpenText eDirectory. It must be returned to the requestor in an out-of-band fashion.

Validate Certificates

OpenText Certificate Server allows you to check the validity of any certificate in the OpenText eDirectory tree. The certificate validation process checks each certificate in the certificate chain back to the trusted root certificate and returns a status of Valid or Invalid.

- To check the validity of certificates for the Organizational CA, see “Validating the Organizational CA's Certificates” on page 658.
- To check the validity of certificates for a server, see “Validating a Server Certificate” on page 668.
- To check the validity of certificates for a user, see “Validating a User Certificate” on page 674.
- To check the validity of certificates for a Trusted Root, see “Validating a Trusted Root Object” on page 682.

Certificates are considered valid if they pass a predefined set of criteria including whether the current time is within the validity period of the certificate, whether it has not been revoked, and whether it has been signed by a CA that is trusted.

When validating user certificates or intermediate CA certificates in CN=trusted roots.CN=security signed by external CAs, the external CA’s certificate must be stored in a Trusted Root object in order for the certificate validation to be successful.

Manage Certificate Revocation Lists

A Certificate Revocation List (CRL) is a published list of revoked certificates and the reason the certificates were revoked.

OpenText Certificate Server provides a system for managing CRLs. This is an optional system, but it must be implemented if you want to be able to revoke certificates created by the Organizational CA. For more information on managing CRLs, see “Certificate Revocation List (CRL) Tasks” on page 683.

During the Certificate Server installation, a CRL container is created if the user has the appropriate rights to create it. If not, the CRL container can be created manually by someone with the appropriate rights after the installation is completed.

A CRL Configuration object can be created in the CRL container. The object contains the configuration information for the CRL objects that are available in the OpenText eDirectory tree. Normally, you have only one CRL Configuration object in your tree. You might need multiple CRL Configuration objects if you are creating or rolling over a new Organizational CA, but only one CRL Configuration object can be used to create new certificates.

A CRL object, also known as a distribution point, can be created in any container in the OpenText eDirectory tree. However, OpenText CRL objects usually reside in a CRL container. A CRL object is automatically created for you when you create a CRL Configuration object. The CRL object contains a CRL file, which contains the detailed CRL information. For a OpenText CRL object, the CRL file is automatically created and updated whenever the server issues a new one. For other CRL objects, you must import a CRL file from a third-party CA. When a server that holds Organization CA is upgraded to OpenText eDirectory 9.2, the upgrade process automatically creates CRL distribution points. Also, OpenText eDirectory provides separate CRL configuration objects for RSA and ECDSA certificates.

Deleting a CRL Configuration object is possible, but it is not recommended. When a CRL Configuration object is deleted, the server quits creating the CRL files. If a CRL file already exists in the location specified in the CRL object, certificate validation continues to use it until it expires. After it expires, all certificates that have a CRL distribution point that references that CRL file fail validation.

If you delete a CRL object, it is re-created the next time the server generates the CRL file. If you delete a CRL object that you created using OpenText Identity Console and import it, then it is gone permanently and any certificates that reference it are considered invalid.

The general rule is to not delete a CRL container, CRL Configuration object, CRL object, or CRL file until one issue date after the last certificate that contains a related distribution point has expired.

Export Private Keys and Certificates

User, server, and CA keys can be marked as exportable when they are created. If a key is exportable, it can be extracted and put in a file along with the associated certificate. The file is written in an industry standard format (PFX or PKCS #12), which allows it to be transported to other platforms. It is encrypted with a user-specified password to protect the private key.

Exporting private keys and certificates can be done to obtain a backup copy of the key, to move the key to a different server, or to share the key between servers.

For more information on exporting private keys and certificates, see “Exporting a User Certificate and Private Key” on page 673.

Import Private Keys and Certificates

You can choose to import a key rather than create a new one at the time a server certificate, a user certificate, or a CA object is created. The key and its associated certificates must be in PFX or PKCS #12 format.

You might choose to import a key rather than create a new one for a CA object to recover from a server failure, to move the Organizational CA from one server to another, or for a CA that is subordinate to another CA.

You might choose to import a user certificate or private key if it has been signed by a third-party CA.

You might choose to import a key rather than create a new one for a Server Certificate object to recover from a server failure, to move the key and certificate to another server, or to share the key and certificate with another server.

Create an SAS Service Object

The SAS service object facilitates communication between a server and its server certificates. If you remove a server from an OpenText eDirectory tree, you also need to delete the SAS service object associated with that server. If you want to put the server back into the tree, you must create the SAS service object to go with that server. If you do not, you cannot create new server certificates.

The SAS service object is automatically created as part of the server health check. You should not need to create it manually.

You can create a new SAS service object only if there is not a properly named SAS service object in the same container as the server object. For example, for a server named WAKE, you will have a SAS service object named SAS Service - WAKE. The utility adds the DS pointers from the Server object to the SAS object, and from the SAS object to the Server object, as well as set up the correct ACL entries on the SAS service object.

If a SAS service object already exists with the proper name, you cannot create a new one. The old SAS service object's DS pointers might be wrong or missing, or the ACLs might not be correct. In this case, you can delete the corrupt SAS service object and use Identity Console to create a new one. If there are server certificates that belong to this server, you need to link them to the SAS service object manually by using the **Other** tab.

For more information on creating a SAS service object, see "Creating an SAS Service Object" on page 650.

NICI

NICI is the underlying cryptographic infrastructure that provides the cryptography for OpenText Certificate Server, NetIQ Modular Authentication Services (NMAS), and other applications.

NICI must be installed on the server in order for OpenText Certificate Server to work properly. NICI does not ship with OpenText Certificate Server. In most cases NICI is provided and installed when OpenText Certificate Server is bundled with another product, such as Open Enterprise Server (OES) or OpenText eDirectory. If you need a newer version of NICI, you can download it from the Software License and Download portal.

Setting Up OpenText Certificate Server

After you install OpenText Certificate Server, you must set it up for use on your network by completing the following tasks:

- "Deciding Which Type of Certificate Authority to Use" on page 641
- "Creating an Organizational Certificate Authority Object" on page 642
- "Subordinate Certificate Authority" on page 644
- "Restrictions for Creating a Certificate Authority Object" on page 646
- "Configuring the Certificate Authority in Suite B Mode" on page 647
- "Creating a Server Certificate Object" on page 647
- "Configuring Cryptography-Enabled Applications" on page 648
- "Additional Components to Set Up" on page 649

Deciding Which Type of Certificate Authority to Use

OpenText Certificate Server allows you to create certificates for both servers and end users. Server certificates can be signed by either the Organizational CA or by an external or third-party CA. User certificates can be signed only by the Organizational CA; however, you can import user certificates signed by a third-party CA in PKCS#12 format.

During the Server Certificate object creation process, you are asked which type of certificate authority will sign the Server Certificate object.

The Organizational Certificate Authority is specific to your organization and uses an organizational-specific public key for signing operations. The private key is created when you create the Organizational Certificate Authority.

A third-party certificate authority is managed by a third party outside of the OpenText eDirectory tree. An example of a third party certificate authority is VeriSign.

Both types of certificate authorities can be used simultaneously. Using one type of certificate authority does not preclude the use of the other.

- "Benefits of Using an Organizational Certificate Authority Provided with OpenText Certificate Server" on page 641
- "Benefits of Using an External Certificate Authority" on page 642

Benefits of Using an Organizational Certificate Authority Provided with OpenText Certificate Server

- **Compatibility.** The Organizational Certificate Authority is compatible with OpenText applications such as LDAP services. Certificates issued by the Organizational CA are X.509 v3 compliant and can also be used by third-party applications.
- **certificate authorityCost savings.** The Organizational Certificate Authority lets you create an unlimited number of public key certificates at no cost; obtaining a single public key certificate through an external Certificate Authority might cost a significant amount of money.

- **Component of a complete and compatible solution.** By using the Organizational Certificate Authority, you can use the complete cryptographic system built into OpenText eDirectory without relying on any external services. In addition, OpenText Certificate Server is compatible with a wide range of OpenText products.
- **Certificate attribute and content control.** An Organizational Certificate Authority is managed by the network administrator, who decides on public key certificate attributes such as certificate life span, key size, and signature algorithm.
- **Simplified management.** The Organizational Certificate Authority performs a function similar to external certificate authorities but without the added cost and complexity.

Benefits of Using an External Certificate Authority

- **Liability.** An external certificate authority might offer some liability protection if, through the fault of the certificate authority, your private key was exposed or your public key certificate was misrepresented.
- **Availability.** An external certificate authority's certificate might be more widely available and more widely trusted by applications outside of OpenText eDirectory.

Creating an Organizational Certificate Authority Object

By default, the OpenText Certificate Server installation process creates the Organizational Certificate Authority (CA) for you. You are prompted to specify an Organizational CA name. When you click **Finish**, the Organizational CA is created with the default parameters and placed in the Security container.

If you want more control over the creation of the Organizational CA, you can create the Organizational CA manually by using OpenText Identity Console. Also, if you delete the Organizational CA, you need to re-create it.

During the creation process, you are prompted to name the Organizational Certificate Authority object and to choose a server to host the Organizational CA service (the server the Organizational CA service will run on). In determining the server to host the Organizational CA service, consider the following:

- Select a server that is physically secure.

  Physical access to the CA server is an important part of the security of the system. If the CA server is compromised, all certificates issued by the CA are also compromised.
- Select a server that is highly available, stable, and robust.

  If the CA service is not available, certificates cannot be created. This affects installation of new servers because certificates need to be created during install.
- Select a server that only runs software you trust.

  Running unknown or questionable software might compromise the CA service.
- Select a server that will not be removed from the tree.

  If the server is removed from the tree, you need to either re-create the CA object by using a backup you made before removing the CA, or you need to create a new CA. If you create a new CA, you might need to replace your existing server and user certificates.
- Select a server that runs a protocol that is compatible with other servers in your tree.

Examples are IP.

- Create an Organization CA with ECDSA certificate.

To create the Organizational Certificate Authority object:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** tile > **CA Management** tile > **OpenText Certificate Server** > **Configure Certificate Authority**.

   If no Organizational Certificate Authority object exists, this opens the Create an Organizational Certificate Authority Object dialog box and the corresponding wizard that creates the object. Follow the prompts to create the object. For specific information on the dialog box or any of the wizard pages, click Help.

   NOTE: Ensure that the CRL file path which is specified here, is in respect with the OpenText eDirectory installation path.

   ---
4. After you have finished creating the Certificate Authority, we recommend that you make a backup of the CA's public/private key pair and store this in a safe and secure place. See "Backing Up an Organizational CA" on page 655.

NOTE: You can have only one Organizational CA for your OpenText eDirectory tree.

OpenText eDirectory allows the administrator to specify the RSA key size, Elliptic Curve and certificate to be used when the default server certificates are generated. These parameters can be specified using the following three attributes on the CA object:

- **ndspkiDefaultRSAKeySize:** Specify the key size for the RSA server certificates. You can specify up to 8192 bit RSA encryption in this field.

  IMPORTANT: Using an X.509 certificate with 8K bit RSA public keys for establishing TLS connection impacts the performance of your OpenText eDirectory servers. OpenText does not recommend configuring OpenText eDirectory servers to use RSA certificates with 8K bit keys because TLS session establishment can be computation intensive for the servers and it may cause significant system slowdown when TLS sessions are established concurrently.

- **ndspkiDefaultECCurve:** Specify the curve for the EC limit for the server certificates. You can specify any one of the following EC:
  - P256
  - P384
  - P521
- **ndspkiDefaultCertificateLife:** Specify the certificate life for the default server certificates. You can specify the server certificate life in years. For example, if you specify 4 in this field, your server certificate life will be set to 4 years. Certificate server ensures that the default server certificates have a minimum validity of 1 year and maximum validity does not exceed the CA's expiry date.

NOTE: ◆`ndspkiDefaultCertificateLife` attribute is only applicable for server certificates.

- If you pass the old default values while configuring a new OpenText eDirectory server, the above mentioned parameters remain unset.
- The above parameters do not affect the existing default certificates. While upgrading the OpenText eDirectory server to 9.2 after specifying these parameters on the CA, the existing default server certificates are not re-created.

If you specify these parameters while configuring a new OpenText eDirectory tree, the Organizational CA certificates are also created with these parameters. For more information, see the OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide.

Subordinate Certificate Authority

OpenText Certificate Server has added support for a Subordinate Certificate Authority. This feature allows the Organizational CA to be subordinate to either a third-party CA or to a CA in another OpenText eDirectory tree. You still can have only one Organizational CA in your OpenText eDirectory tree.

The following are some of the reasons to have a Subordinate CA:

- Allows the Organizational CA to become part of an existing third-party PKI
- Allows multiple trees to share a common PKI Trusted Root (or Trust Anchor)
- Allows for greater security of the Root CA by having the CA reside on a more secure system
- Provides less risk by having the Root CA reside in a tree that is more tightly managed (for example, in a tree protected from rogue-administrators/users)

- "Creating a Subordinate Certificate Authority" on page 644
- "Creating PKCS#12 Files for a Subordinate CA" on page 644

Creating a Subordinate Certificate Authority

In order to create a Subordinate CA, you must first delete the existing Organizational CA (see "Deleting the Organizational CA" on page 659). You must already have a PKCS#12 file containing the public/private keys and the certificate chain for the Subordinate CA. You can either obtain this file directly from a third-party CA or use "Creating PKCS#12 Files for a Subordinate CA" on page 644 to learn how to create one. In order to create the Subordinate CA, connect to the tree in OpenText Identity Console and use the Configure Certificate Authority task, using the Import creation method.

Creating PKCS#12 Files for a Subordinate CA

1 Create a Server Certificate object (or KMO) and a PKCS#10 CSR with ECDSA and RSA keys.

1a Launch OpenText Identity Console.

1b On the OpenText Identity Console home page > **Server Certificate Management** tile > **Create Server Certificate** +.

1c Select the server that will eventually host the CA, specify a certificate nickname, select the Custom creation method, then click **Next**.

1d Select **External Certificate Authority**, then click **Next**.

1e Select the algorithm and a key size and make sure that **Allow Private Key to Be Exported** is selected, then click **Next**.

IMPORTANT: If you want to use both RSA and ECDSA certificates in your OpenText eDirectory environment, repeat this step for the certificate that you want to use. OpenText recommends that you use a key size of 2048 bits for RSA and 384 bits for ECDSA.

1f Click the **Edit** button to the right of the **Subject name** field and edit the **Subject name** to reflect the subordinate CA and tree, select the Signature algorithm (OpenText recommends that you use a stronger algorithm than SHA-1).

1g Verify that the summary is correct, then click **Finish**.

1h Click **Save Certificate Signing Request**, then follow the prompts to save the CSR to a file.

2 Get the CSR signed to create a certificate.

2a If the Subordinate CA is to be part of a third-party PKI, have the third-party CA create the certificate from the CSR.

or

If the Subordinate CA is to be signed by a CA in another OpenText eDirectory tree, continue with Step 2b.

NOTE: In case of ECDSA, the CSR can be signed only by ECDSA CA.

2b Launch OpenText Identity Console.

2c On the OpenText Identity Console home page > **Server Certificate Management** tile > **Create Server Certificate** +.

2d Select the file containing the CSR.

2e Select a key type of **Certificate Authority**, deselect **Enable Extended Key Usage**.

2f Select the Certificate Authority Certificate type and then select either the **Unspecified** or a **Specific Path** length.

2g Click **Certificate Parameters** drop down > Verify the subject name and edit it if necessary. Specify a validity period (5-10 years is recommended).

2h Click **Certificate Format** drop down > Select a format for the certificate, click **Next**.

2i Click **Finish**.

3 Acquire the CA certificates.

3a If the Subordinate CA is to be part of a third-party PKI, acquire the CA certificates from the third party.

or

If the Subordinate CA is to signed by a CA in another OpenText eDirectory tree, continue with Step 3b.

3b Launch Identity Console.

3c On the OpenText Identity Console home page > click **Server Certificate Management** tile > click **CA Management**.

3d Select **Self-Signed Certificate ECDSA**.

Selecting the checkbox **Export Private key** enables export format as PKCS12.

3e Click **OK** > click **Save the exported file** option and save the file to the required location > click **OK**.

4 Import the certificates into the Server Certificate object (or KMO).

4a Launch OpenText Identity Console.

4b On the OpenText Identity Console home page > **CA Management** tile.

4c On the **Create Certificate Authority** page > select the **Server** > provide **Object Name** > choose a **Creation method** > click **Next** > click **OK**.

4d Choose **Extended Key Usage Specification** drop down > select the required authority.

4e Click **Basic Constrains** drop down > select as **Specified**.

4f Click **Next** > click **OK**.

4g Select the Server Certificate object (or KMO) created in Step 1, then click **Import**.

4h Select the two files containing the certificates acquired in Step 2 and Step 3, then click **OK**.

IMPORTANT: If you want to use both RSA and ECDSA certificates in your OpenText eDirectory environment, repeat this step for the certificate that you want to use.

---

5 Export the public/private keys to a PKCS#12 file.

5a Continuing from Step 4h, click **Export**, choose to include the private key, then click **Next**.

5b Click **Save the Exported Certificate to a File**, then follow the prompts to save the PKCS#12 file.

5c Make a copy of this file and store it in a secure place along with the password.

6 (Optional) Delete the Server Certificate object (or KMO).

7 Delete the Organizational CA. For more information, see “Deleting the Organizational CA” on page 659.

8 Import the sub CA certificates and private keys from the PKCS#12 file. For more information, “Restoring an Organizational CA” on page 656.

Restrictions for Creating a Certificate Authority Object

OpenText eDirectory 9.0 and above supports both RSA and EC certificates for the CA. The following considerations apply for creating a CA object with RSA and EC certificates:

- The subject name of RSA and EC certificates must be same.
- RSA and EC CA must not be in a mixed mode. You can have RSA and EC CA either as a root CA or SubCA. For example, you cannot have RSA CA as a root CA and EC CA as a SubCA or vice versa.
- OpenText eDirectory does not allow you to import self-signed CA certificates generated by a third party application outside of OpenText eDirectory.

Configuring the Certificate Authority in Suite B Mode

Before configuring the CA in the Suite B mode, ensure that the server where CA is hosted has NICI 3.0 installed and is configured to run Enhanced Background Authentication.

To configure the CA to operate in the Suite B mode, perform the following steps:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.
3. go to **Certificate Management** tile > **CA Management**.
4. Click CA Configuration  and select **Enable Suite B Mode** checkbox.
5. Click **Apply**.

When the CA is in Suite B mode, the Certificate Server does not allow you to create RSA certificates. For information about which ECDSA certificates are supported, see “OpenText Certificate Server Components” on page 634.

If you add a server with an older version of OpenText eDirectory to an OpenText eDirectory 9.0 or above tree configured with Suite B, the Certificate Server does not create certificates for the server because the server does not have the Suite B capability. To enable the Suite B capability on these servers, you must upgrade them to OpenText eDirectory 9.0 or above.

Creating a Server Certificate Object

Server Certificate objects are created in the container that holds the server's OpenText eDirectory object. Depending on your needs, you might create a separate Server Certificate object for each cryptography-enabled application on the server, or you might create one Server Certificate object for all applications used on that server.

NOTE: The terms Server Certificate object and Key Material object (KMO) are synonymous. The schema name of the OpenText eDirectory object is NDSPKI:Key Material.

When you install Certificate Server, OpenText eDirectory automatically creates the Server Certificate object with the default parameters and places it in the container where the target server resides. If you ever need to overwrite or create new default certificates, you can use the Create Default Certificates Wizard. See “Creating Default Server Certificate Objects” on page 661.

If you want more control over the creation of the Server Certificate object, you can create the Server Certificate object manually. You can also create additional Server Certificate objects.

- “Manually Creating a Server Certificate Object” on page 647
- “Hints for Creating Server Certificates” on page 648

Manually Creating a Server Certificate Object

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see “Entry Rights Needed to Perform Tasks” on page 702.

3 go to **Certificate Management** and click **Server Certificate Management**.

4 Click  to create a server certificate object. This opens the **Create a Server Certificate** page.

5 Specify the **Nickname** of the server certificate object being created.

6 Select the **Creation Method** and click **Next**.

7 On the **Summary** tab, view the parameters that apply to the User Certificate being created

8 Click **OK**.

Hints for Creating Server Certificates

During the Server Certificate object creation process, you are prompted to name the key pair and choose the server that the key pair will be associated with. The Server Certificate object is generated by OpenText Certificate Server, and its name is based on the key pair name that you choose.

If you choose the Custom creation method, you are also prompted to specify whether the Server Certificate object will be signed by your organization's Organizational Certificate Authority or by an external certificate authority. For information about making this decision, see “Deciding Which Type of Certificate Authority to Use” on page 641.

If you decide to use your organization's Organizational CA, the server that the Server Certificate object is associated with must be able to communicate with the server that hosts the Organizational CA, or it must be the same server. These servers must be running the same protocol (IP).

If you decide to use an external certificate authority to sign the certificate, the server that the Server Certificate object is associated with generates a certificate signing request that you need to submit to the external certificate authority.

After the certificate is signed and returned to you, you need to install it into the Server Certificate object, along with the trusted root for the external Certificate Authority.

After you have created the Server Certificate object, you can configure your applications to use it. (See “Configuring Cryptography-Enabled Applications” on page 648.) Keys are referenced in the application's configuration by the key pair name that you entered when you created the Server Certificate object.

Configuring Cryptography-Enabled Applications

After you have configured OpenText Certificate Server, you must configure your individual cryptography-enabled applications so that they can use the custom certificates that you created. The configuration procedures are unique to the individual applications, so we recommend that you consult the application's documentation for specific instructions.

Additional Components to Set Up

OpenText Certificate Server includes some additional components that can be set up to provide additional functionality.

- "Creating a User Certificate" on page 649
- "Creating a Trusted Root Container" on page 649
- "Creating a Trusted Root Object" on page 650
- "Creating an SAS Service Object" on page 650

Creating a User Certificate

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > click **Certificate Management** > **User Certificate Management** > **User Certificate Management** > **Create User Certificate** +.

   This opens a wizard that helps you create the user certificate. Follow the prompts to create the object. For specific information on the wizard pages, click **Help**.

Creating a Trusted Root Container

You can create a Trusted Root container anywhere in the OpenText eDirectory tree.

1. Launch OpenText Identity Console.
2. Log in to theOpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. go to **Certificate Management** and click **Trusted Root Management**.
4. Click + to create a new Trusted Root Container.
5. Specify the **Name** and **Context** for the Trusted Root container. You can select the **Context** for the Trusted Root container using object selector.
6. Click **OK**.

NOTE: Different applications might require that the Trusted Root container be given a specific name and be in a specific location in the OpenText eDirectory tree. OpenText Certificate Server requires that the Trusted Root container be named Trusted Roots and be located in the Security container. The certificates in this container are used to validate user certificates signed by external CAs and intermediate CA certificates stored in Trusted Root objects. Server certificates and the Organizational CA's certificates use the certificate chain stored in their own objects.

Creating a Trusted Root Object

A Trusted Root object can only reside in a Trusted Root container.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > click **Certificate Management** > **Trusted Root Management** > **Trusted Roots** tab > **Create Trusted Root** +.

   This opens the Create a Trusted Root Object Wizard that helps you create the Trusted Root object. Follow the prompts to create the object. For specific information on the wizard pages, click **Help**.

NOTE: Any type of certificate can be stored in a Trusted Root object (CA certificates, intermediate CA certificates, or user certificates).

Creating an SAS Service Object

The SAS Service object is automatically created as part of the server health check. You should not need to create it manually. If you need to create it manually, use the following procedure:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > click **Certificate Management** > **SAS Service Object** > **Create a SAS Service Object** +.

   This opens the Create a SAS Service Object Wizard that helps you create the SAS Service object.

   Follow the prompts to create the object. For specific information on the wizard pages, click ⓘ.

Managing OpenText Certificate Server

As a system administrator, you need to perform several tasks to maintain the public key cryptography services provided through OpenText Certificate Server. Use Identity Console to perform these tasks. This section provides a brief overview and specific information on completing each task.

Certificate Authority tasks:

- "Creating an Organizational Certificate Authority Object" on page 653
- "Issuing a Public Key Certificate" on page 653
- "Viewing the Organizational CA's Properties" on page 654
- "Viewing an Organizational CA's Public Key Certificate Properties" on page 654
- "Viewing the CA's Self-Signed Certificate Properties" on page 654

- “Exporting the Organizational CA's Self-Signed Certificate” on page 655
- “Backing Up an Organizational CA” on page 655
- “Restoring an Organizational CA” on page 656
- “Moving the Organizational CA to a Different Server” on page 658
- “Validating the Organizational CA's Certificates” on page 658
- “Deleting the Organizational CA” on page 659
- “Rolling Over an Organizational CA” on page 660

Server Certificate object tasks:

- “Creating Server Certificate Objects” on page 661
- “Creating Default Server Certificate Objects” on page 661
- “Importing a Public Key Certificate into a Server Certificate Object” on page 662
- “Exporting a Trusted Root or Public Key Certificate” on page 663
- “Deleting a Server Certificate Object” on page 664
- “Viewing a Server Certificate Object's Properties” on page 665
- “Viewing a Server Certificate Object's Public Key Certificate Properties” on page 665
- “Viewing a Server Certificate Object's Trusted Root Certificate Properties” on page 665
- “Backing Up a Server Certificate Object” on page 666
- “Restoring a Server Certificate Object” on page 667
- “Server Certificate Objects and Clustering” on page 668
- “Validating a Server Certificate” on page 668
- “Revoking a Trusted Root or Self Signed Certificate” on page 669
- “Moving a Server Certificate Object to a Different Server” on page 669
- “Replacing a Server Certificate Object's Keying Material” on page 670

User Certificate tasks:

- “Creating User Certificates” on page 671
- “Creating User Certificates in Bulk” on page 671
- “Importing a Public Key Certificate into a User Object (with or without the Private Key)” on page 671
- “Viewing a User Certificate's Properties” on page 672
- “Exporting a User Certificate” on page 672
- “Exporting a User Certificate and Private Key” on page 673
- “Validating a User Certificate” on page 674
- “Revoking a User Certificate” on page 675
- “Deleting a User Certificate and Private Key” on page 675

X.509 Certificate Self-Provisioning:

- “Overview” on page 676
- “User Self-Provisioning” on page 676

- “Server Self-Provisioning” on page 677
- “Certificate Self-Provisioning and the Issue Certificate Task” on page 678

Using OpenText eDirectory Certificates with External Applications

- “PKI Health Check Functionality” on page 679
- “Configuring the SAS:Service Object to Export eDirectory Certificates” on page 680

Trusted Root object tasks:

- “Create a Trusted Root Container” on page 637
- “Create a Trusted Root Object” on page 638
- “Viewing a Trusted Root Object's Properties” on page 681
- “Validating a Trusted Root Object” on page 682
- “Revoking a Trusted Root Certificate” on page 682

Certificate Revocation List (CRL) Tasks:

- “Creating a CRL Container Manually” on page 683
- “Deleting a CRL Container” on page 684
- “Creating a CRL Configuration Object” on page 684
- “Activating a CRL Configuration Object” on page 685
- “Viewing and Modifying a CRL Configuration Object's Properties” on page 685
- “Deleting a CRL Configuration Object” on page 687
- “Creating a CRL Object” on page 687
- “Exporting a CRL File” on page 688
- “Replacing a CRL File” on page 688
- “Viewing a CRL Object's Properties” on page 689
- “Deleting a CRL Object” on page 690

OpenText eDirectory tasks:

- “Resolving Multiple Security Containers, Organizational CAs, KAP Containers, and W0 Objects” on page 690
- “Restoring or Re-creating a Security Container” on page 691
- “Restoring or Re-creating KAP and W0” on page 691

Application tasks

Certificate Authority Tasks

- “Creating an Organizational Certificate Authority Object” on page 653
- “Issuing a Public Key Certificate” on page 653
- “Viewing the Organizational CA's Properties” on page 654
- “Viewing an Organizational CA's Public Key Certificate Properties” on page 654
- “Viewing the CA's Self-Signed Certificate Properties” on page 654

- “Exporting the Organizational CA's Self-Signed Certificate” on page 655
- “Backing Up an Organizational CA” on page 655
- “Restoring an Organizational CA” on page 656
- “Moving the Organizational CA to a Different Server” on page 658
- “Validating the Organizational CA's Certificates” on page 658
- “Deleting the Organizational CA” on page 659
- “Rolling Over an Organizational CA” on page 660

Creating an Organizational Certificate Authority Object

This task is described in “Creating an Organizational Certificate Authority Object” on page 642.

Issuing a Public Key Certificate

This task allows you to generate certificates for cryptography-enabled applications that do not recognize Server Certificate objects.

Your Organizational CA works the same way as an external CA. That is, it has the ability to issue certificates from certificate signing requests (CSRs). You can issue certificates using your Organizational CA when a user sends a CSR to you for signing. The user requesting the certificate can then take the issued certificate and import it directly into the cryptography-enabled application.

To issue a public key certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see “Entry Rights Needed to Perform Tasks” on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** tile, click **Issue Certificate**.
4. Use the **Chose file** button to locate a CSR file, open the file, then click **Open**.
5. Specify the key type, the key usage, and the extended key usage, then click **Next**.
6. Specify the certificate basic constraints, then click **Next**.
7. Specify the subject name, the validity period, the effective and expiration dates, and any custom extensions, then click **Next**.
8. Review the parameters sheet. If it is correct, click **Finish**. If not, click **Back** until you reach the point where you need to make changes.

   When you click **Finish**, a dialog box explains that a certificate has been created. You can save the certificate to the system clipboard in Base64 format, to a Base64-formatted file, or to a binary DER-formatted file. You can also click **Details** to view details about the issued certificate.

Viewing the Organizational CA's Properties

In addition to the OpenText eDirectory rights and properties that can be viewed with any OpenText eDirectory object, you can also view properties specific to the Organizational CA, including the properties of the public key certificate and the self-signed certificate associated with it.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **CA Management** tile > click **RSA Certificate**.

   This brings up the property pages for the Organizational CA, which include a General page, a Details page, an Extension page, and so on.
4. Click the tabs that you want to view.

Viewing an Organizational CA's Public Key Certificate Properties

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** tile > **CA Management**.

   This brings up the property pages for the Organizational CA, which include a General page, a CRL page, a Certificates page, and other OpenText eDirectory-related pages.
4. Click **Close**.

Viewing the CA's Self-Signed Certificate Properties

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. Go to **Certificate Management** tile > **CA Management** > **Self Signed Certificate ECDSA**.

   This brings up the Certificate Details of selected **Self Signed Certificate** which includes **General**, **Details**, **Extensions, CRL Configuration** and **CRL Information** tabs.
4. If this is a Subordinate CA, there is no self-signed certificate.
5. Click **Close**.

Exporting the Organizational CA's Self-Signed Certificate

The self-signed certificate can be used for verifying the identity of the Organizational CA and the validity of a certificate signed by the Organizational CA.

From the Organizational CA's property page, you can view the certificates and properties associated with this object. From the self-signed certificate property page, you can export the self-signed certificate to a file for use in cryptography-enabled applications.

The self-signed certificate that resides in the Organizational CA is the same as the Trusted Root certificate in a Server Certificate object that has a certificate signed by the Organizational CA. Any service that recognizes the Organizational CA's self-signed certificate as a trusted root can accept a valid user or server certificate signed by the Organizational CA.

This task does not apply if the CA is a Subordinate CA.

To export the Organizational CA's self-signed certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. go to **Certificate Management** tile > **CA Management** and select **Self Signed Certificate**.
4. Click **Export**  to export the certificate.
5. Click **OK**.

Backing Up an Organizational CA

OpenText recommends that you back up your Organizational CA's private key and certificates in case the Organizational CA's host server has an unrecoverable failure. If a failure should occur, you can use the backup file to restore your Organizational CA to any server in the tree.

NOTE: The ability to back up an Organizational CA is available only for Organizational CAs created with Certificate Server version 9.0 at a minimum. In previous versions of Certificate Server, the Organizational CA's private key was created in a way that made exporting it impossible.

The backup file contains the CA's private key, self-signed certificate, public key certificate, and several other certificates necessary for it to operate. This information is stored in PKCS #12 format (also known as PFX).

The Organizational CA should be backed up when it is working properly.

With Certificate Server 9.0 and later, in order to completely back up the Certificate Authority, it is necessary to back up the CRL database and the Issued Certificates database.

For other platforms, both of these databases are located in the same directory as the OpenText eDirectory `dib` files. The defaults for these locations are as follows:

- Windows: `c:\novell\nds\dibfiles`
- Linux: `/var/opt/novell/edirectory/data/dib`

These defaults can be changed at the time that OpenText eDirectory is installed.

The files to back up for the CRL database are `crl.db`, `crl.01` and the `crl.rfl` directory. The files to back up for the Issue Certificates database are `cert.db`, `cert.lck`, `cert.01`, and the `cert.rfl` directory.

The OpenText eDirectory `dib` directory should be part of a standard and regular backup plan.

To back up the Organizational CA:

1. Launch OpenText Identity Console.
2. Login to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** > **CA Management**.
4. Select either the **Self Signed Certificate** or the **Public Key Certificate**. Both certificates are written to the file during the backup operation.

   OpenText recommends that you select the **Self Signed Certificate** for RSA and ECDSA certificates separately.
5. Click **Export** .

   This opens a wizard that helps you export the certificates to a file.
6. Choose to export the private key, specify a password with 6 or more alphanumeric characters to use in encrypting the PFX file, then click **Next**.
7. Click the **Save the exported certificate to a file** link and provide the filename and the location for the backup file.
8. Click **Save**.
9. Click **Close**.

   The encrypted backup file is written to the location specified. It is now ready to be stored in a secure location for emergency use.

IMPORTANT: The exported file should be put on backup media and stored in a secure place. The password used to encrypt the file should be committed to memory or stored in a safe place to ensure that it is available when needed, but inaccessible to others.

Restoring an Organizational CA

If the Organizational CA object has been deleted or corrupted, or if the Organizational CA's host server has suffered an unrecoverable failure, the Organizational CA can be restored to full operation through using a backup file created as described in "Backing Up an Organizational CA" on page 655.

NOTE: If you were unable to make a backup of the Organizational CA, the Organizational CA might still be recovered if NICI 2.*x* is installed on the server and a backup was made of the NICI configuration information.

With Certificate Server 9.0, in order to completely restore the Certificate Authority, it is necessary to restore the CRL database and the Issued Certificates database.

Both of these databases are located in the same directory as the OpenText eDirectory `dib` files. The defaults for these locations are as follows:

- Windows: `c:\novell\nds\dibfiles`
- Linux: `/var/opt/novell/edirectory/data/dib`

These defaults can be changed at the time that OpenText eDirectory is installed.

The files to restore for the CRL database are `crl.db`, `crl.01` and the `crl.rfl` directory. The files to restore for the Issue Certificates database are `cert.db`, `cert.lck`, `cert.01`, and the `cert.rfl` directory.

The OpenText eDirectory `dib` directory should be part of a standard and regular backup plan.

To restore the Organizational CA:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. (Conditional) If the Organizational CA object still exists, you need to delete it:

   3a On the Identity Console home page, click **CA Management**.

   3b Browse to and click the Organizational CA object.

   3c Click **OK**.
4. Click **CA Management** tile > **Server Certificate Management**.

   This opens the Create an Organizational Certificate Authority Object dialog box and the corresponding wizard that creates the object.
5. In the creation dialog box, specify the server that should host the Organizational CA and the name of the Organizational CA object.
6. Select the **Import** option.

   Select both RSA and ECDSA certificates. The Certificate Server requires that both certificates have the same subject name.

   IMPORTANT: The Certificate Server does not support importing external self-signed CA certificates. However, it allows you to import subordinate CA certificates.

   ---
7. Click **Next**.
8. In the dialog that opens, click **Browse** and select the name of the file for RSA and ECDSA.
9. Enter the password used to encrypt the file when the backup was made.
10. Click **OK**.

    The Organizational CA's private key and certificates have now been restored and the CA is fully functional. The file can now be stored again for future use.

IMPORTANT: Be sure to protect your backup media.

Moving the Organizational CA to a Different Server

You can move your Organizational CA from one server to another by using the backup and restore procedures outlined in "Backing Up an Organizational CA" on page 655 and "Restoring an Organizational CA" on page 656.

With Certificate Server 3.2 and later, in order to completely move the Certificate Authority, it is necessary to move the CRL database and the Issued Certificates database.

For other platforms, both of these databases are located in the same directory as the OpenText eDirectory `dib` files. The defaults for these locations are as follows:

- Windows: `c:\novell\nds\dibfiles`
- Linux: `/var/opt/novell/edirectory/data/dib`

These defaults can be changed at the time that OpenText eDirectory is installed.

The files to move for the CRL database are `crl.db`, `crl.01` and the `crl.rfl` directory. The files to move for the Issue Certificates database are `cert.db`, `cert.lck`, `cert.01`, and the `cert.rfl` directory.

1. Make sure the Organizational CA is functional
2. Back up the Organizational CA
3. Export the CA keys
4. Delete the Organizational CA object.
5. Stop OpenText eDirectory on both the servers
6. Copy the `cert` and `crl` files from the source to the destination server including the `rfl` logs
7. Start OpenText eDirectory on both the servers
8. Recreate the Organizational CA on the destination server

IMPORTANT: Be sure to protect your backup media.

Validating the Organizational CA's Certificates

If you suspect a problem with a certificate or think that it might no longer be valid, you can easily validate the certificate by using OpenText Identity Console. Any certificate in the eDirectory tree can be validated, including certificates issued by external CAs.

The certificate validation process includes several checks of the data in the certificate as well as the data in the certificate chain. A certificate chain is composed of a root CA certificate and, optionally, the certificates of one or more intermediate CAs.

A result of Valid means that all certificates in the certificate chain were found to be valid. Certificates are considered valid if they pass a predefined set of criteria including whether the current time is within the validity period of the certificate, whether it has not been revoked, and whether it has been signed by a CA that is trusted. Only those certificates with a CRL distribution point extension or an OCSP AIA extension are checked for revocation.

A result of Invalid means that one or more certificates in the certificate chain were found to be invalid or their validity could not be determined. Additional information is provided for these certificates, indicating which certificate is considered invalid and why. Click **Help** for more information about the reason.

To validate a certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. go to **Certificate Management** and click **CA Management**.
4. Under **Certificates** tab, select a certificate and click validate .
5. The `Validity Results` displays the **Status** of the **Certificate**. If the certificate is not valid, the reason is given.
6. Click **OK**.

Deleting the Organizational CA

Deleting the Organizational CA object should be done only if absolutely necessary or if you are restoring the Organizational CA from a backup (see "Restoring an Organizational CA" on page 656). The only safe way to delete the object is to do a backup first so that it can be restored later.

However, there are times when the Organizational CA must be deleted and not restored. For example, when merging trees, only one Organizational CA can be in the resulting tree; the other CA must be deleted. Or, when the Organizational CA's host server is irreparably damaged and no backup of the CA or the NICI configuration was made, the only option remaining is to delete the CA and to begin again.

To delete the Organizational CA object:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. Back up the self-signed certificate without the private key.
4. Create a Trusted Root certificate using the self-signed certificate in the CN=trusted roots.CN=security container. For more information, see
5. On the OpenText Identity Console home page, click **Certificate Management** tile > select the certificate that you want to delete > click **Delete Object** .
6. Click **OK**.

Rolling Over an Organizational CA

Two important issues that must be considered when replacing the Organizational Certificate Authority (CA) certificates are:

- The types of certificates that are being managed
- The reason that the CA is being replaced

Server Certificate objects (KMOs) contain both the public key certificate for the server and the Trusted Root certificate with which the public key certificate was signed.

User certificates are stored as attributes on the user object and are not paired with the Trusted Root that signed them. Therefore, when the trusted root certificate is replaced, server certificates are still valid because the trusted root is still accessible. However, user certificates are immediately invalid unless the Trusted Root certificate is placed in the Trusted Roots container where certificate validation can find it.

There are three reasons to replace the CA:

- The CA has reached the end of its validity (the CA is expiring).
- The CA has been compromised.
- You want to replace the CA certificate for some other reason (a stronger key is desired, a new security policy has made it necessary, you want to have an externally signed CA, etc.).

If the CA is expiring, the certificates that the CA signed are also going to expire. After replacing the CA, each of the signed certificates should be re-created with the new CA.

If the CA has been compromised, then replacing the CA invalidates the user certificates that were signed by the old CA. You can easily replace them by running the Create Default Certificates task in OpenText Identity Console. All certificates that are created by default by Certificate Server are re-created with the new CA. Any certificates that were created in a custom manner need to be manually re-created with the new CA. For more information on creating default certificates through, see “Creating Default Server Certificate Objects” on page 661.

If you want to re-create the CA for some other reason, then storing the trusted root certificate in the Trusted Roots container keeps user certificates valid until you have a chance to re-create them at your convenience.

To replace the trusted root certificate:

1. Back up the current CA in case you want to recover it later.
2. Export the Trusted Root certificate that has been used to create the certificates. In older systems, this is most likely the self-signed certificate.

   Recently, the ability has been added to externally sign the CA certificate. If the CA is externally signed, export the public key certificate. All certificates in the chain must have their own object in the Trusted Roots container.

   If the CA has not been compromised, create a Trusted Root certificate in the Trusted Roots container. This ensures that user certificates are still valid until they can be replaced.
3. Delete the old CA. For information on deleting the Organizational CA, see “Deleting the Organizational CA” on page 659.
4. Create a new CA. For information on creating a new Organizational CA, see “Creating an Organizational Certificate Authority Object” on page 653.

5 If necessary, re-create server certificates by using the Create Default Certificate task in OpenText Identity Console. For information on creating default certificates through OpenText Identity Console, see “Creating Default Server Certificate Objects” on page 661.

   Re-create other server certificates that are not generated by default.

6 If necessary, re-create user certificates by using the Create User Certificate task in OpenText Identity Console or by viewing the user properties, viewing certificates, and clicking **New**.

Server Certificate Object Tasks

- “Creating Server Certificate Objects” on page 661
- “Creating Default Server Certificate Objects” on page 661
- “Importing a Public Key Certificate into a Server Certificate Object” on page 662
- “Exporting a Trusted Root or Public Key Certificate” on page 663
- “Deleting a Server Certificate Object” on page 664
- “Viewing a Server Certificate Object's Properties” on page 665
- “Viewing a Server Certificate Object's Public Key Certificate Properties” on page 665
- “Viewing a Server Certificate Object's Trusted Root Certificate Properties” on page 665
- “Backing Up a Server Certificate Object” on page 666
- “Restoring a Server Certificate Object” on page 667
- “Server Certificate Objects and Clustering” on page 668
- “Validating a Server Certificate” on page 668
- “Revoking a Trusted Root or Self Signed Certificate” on page 669
- “Moving a Server Certificate Object to a Different Server” on page 669
- “Replacing a Server Certificate Object's Keying Material” on page 670

Creating Server Certificate Objects

This task is described in “Creating a Server Certificate Object” on page 647.

Creating Default Server Certificate Objects

The Certificate Server installation creates default Server Certificate objects.

- SSL CertificateDNS - *server_name*
- A certificate for each IP address configured on the server (IPAG*xxx.xxx.xxx.xxx* - *server_name*)
- A certificate for each DNS name configured on the server (DNSAG*www.example.com* - *server_name*)

NOTE: OpenText eDirectory does not automatically create SSL CertificateIP. SSL Certificate DNS contains all the IPs listed in the Subject Alternative Name. When you attempt to create or repair the default certificates using the PKI OpenText Identity Console application, the SSL CertificateIP

certificate is not created or repaired by default. However, the OpenText Identity Console's Certificate Management tile provides a check box that you can select to override the default behavior and force the creation/repair of the SSL CertificateIP certificate.

OpenText eDirectory 9.0 and above automatically creates ECDSA certificates if Organization CA has a ECDSA certificate.

If these certificates become corrupt or invalid for some reason, or if you just want to replace the existing default certificates, you can use the Create Default Server Certificates Wizard, as described in the following procedure:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. go to **Certificate Management** tile and click **Default Certificates**.
4. Select the server or servers that you want to create default certificates for, then click **Next**.
5. Select **Yes** if you want to overwrite the existing default server certificates or select **No** if you want to overwrite the existing default server certificates only if they are invalid.
6. (Single Server only) If you want to use the existing default IP address, select that option. If you want to use a different IP address, select that option and specify the new IP address.
7. (Single Server only) If you want to use the existing DNS address, select that option. If you want to use a different DNS address, select that option and specify the new DNS address.
8. Click **Next**.
9. Review the parameters summary that are applicable for the default certificate and then click **Finish**.

If you want more control over the creation of the Server Certificate object, you can create the Server Certificate object manually. For more information, see "Manually Creating a Server Certificate Object" on page 647.

Importing a Public Key Certificate into a Server Certificate Object

You import a public key certificate after you have created a certificate signing request (CSR) and the Certificate Authority (CA) has returned the signed public key certificate to you. This task applies when you have created a Server Certificate object by using the Custom option with the External CA signing option.

There are several ways in which the CA can return the certificate. Typically, the CA either returns one or more files each containing one certificate, or returns a file with multiple certificates in it. These files can be binary, DER-encoded files (`.der`, `.cer`, `.crt`., `.p7b`) or they can be textual, Base64-encoded files (`.cer`, `.b64`).

If the file has multiple certificates in it, it must be in PKCS #7 format in order to be imported into a Server Certificate object. Additionally, the file must contain all of the certificates to be imported into the object (the root-level CA certificate, any intermediate CA certificates, and the server certificate).

If the CA returns multiple files to you as a result of signing the certificate, each file contains a different certificate that must be imported into the Server Certificate object. If there are more than two files (one for the root-level CA, one or more for the intermediate CAs, and one for the server certificate), these files must be combined into a PKCS #7 file in order to be imported into a Server Certificate object.

There are several ways to create a PKCS #7 file. One way is to import all of the certificates into Internet Explorer. After they have been imported, the server certificate and all of the certificates in the certificate chain can be exported in PKCS #7 format by using Internet Explorer. For more information on how to do this, see "External CAs" on page 830.

Some CAs do not return a root-level CA certificate along with the server certificate. In order to obtain the root-level CA certificate, contact the CA provider directly or call Technical Support.

To import the certificates into a Server Certificate object:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificates Management**.
4. Click **Import** next to the Server Certificate object you want to modify.
5. Browse for and select the certificate data file.
6. Browse for and select the trusted root data file.

   If all certificates are contained in a single file, leave this field blank.
7. Click **OK**.

Exporting a Trusted Root or Public Key Certificate

You export a certificate to a file for the following reasons:

- A client (such as an Internet browser) can use it to verify the certificate chain sent by a cryptography-enabled application.
- To provide a backup copy of the file.

You can export the certificate in two file formats: DER-encoded (`.der`) and Base64-encoded (`.b64`). The `.crt` extension can also be used for DER-encoded certificates. You can also export to the system clipboard in Base64 format so that the certificate can be pasted directly into a cryptography-enabled application.

To export a trusted root or public key certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as a user with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the home page of OpenText Identity Console, click **Certificate Management** tile > **Server Certificates**.

4 Select the Server Certificate object the particular application is configured to use.

5 Click **Export Server Certificate**.

This opens a wizard that helps you export the certificate to a file.

6 Use the drop-down list to specify which certificate to export.

7 Choose whether to Export private key > provide password > retype password.

8 Select an export format (binary DER or text encoded base64), then click **OK**.

9 Click **Save the exported certificate to a file** and save the file to a location of your choice.

10 Click **OK**.

11 Use the file as needed.

For example, if you want to install a trusted root certificate in an Internet Explorer browser, double-click the file. This initiates a wizard that will accept the CA as a trusted root. Accepting the CA as a trusted root means that the browser automatically accepts SSL connections with services that use certificates issued by this CA.

Deleting a Server Certificate Object

You should delete a Server Certificate object if you suspect that the private key has been compromised, if you no longer want to use the key pair, or if the trusted root in the Server Certificate object is no longer trusted.

IMPORTANT: After the Server Certificate object is deleted, you cannot recover it unless you have previously made a backup. Before you delete this object, make sure that no cryptography-enabled applications still need to use it.You can re-create a Server Certificate object, but you will need to reconfigure any applications that referenced the old object.

To delete a Server Certificate object:

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see “Entry Rights Needed to Perform Tasks” on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificate**.

4 Select the Server Certificate object you want to delete.

5 Click **Delete** > **OK**.

Viewing a Server Certificate Object's Properties

In addition to the OpenText eDirectory rights and properties that are viewable with any OpenText eDirectory object, you can also view properties specific to the Server Certificate object, including the properties of the public key certificate and the Trusted Root certificate associated with it, if they exist.

To view a Server Certificate object's properties:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificates Management**.
4. Click the nickname of the Server Certificate object you want to view.
5. Server Certificate Object's Properties such as General, Details, Extensions can be viewed.
6. Click **Cancel**.

Viewing a Server Certificate Object's Public Key Certificate Properties

To view a Server Certificate object's public key certificate properties:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as a user with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** > **Server Certificate Management**.
4. Select the Server Certificate object you want to view. > click **View Chain**.
   - If a public key certificate is installed, the property page displays the subject's fully typed name, the issuer's fully typed name, and the validity dates of the public key certificate.
   - If the public key certificate has not yet been installed, the property page indicates this.
5. To view additional information about a public key certificate, click the certificate's nickname to view the Details page.

   The Details page has information contained in the public key certificate.
6. Click **Close**.

Viewing a Server Certificate Object's Trusted Root Certificate Properties

To view a Server Certificate object's Trusted Root certificate properties:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **Trusted Root Management**.

4 Browse to and select the Server Certificate object you want to view.

5 Click **OK**.

6 Click **Trusted Root Certificate**.

- If a Trusted Root certificate is installed, the property page displays the subject's fully typed name, the issuer's fully typed name, and the validity dates of the trusted root certificate.
- If the Trusted Root certificate has not yet been installed, the property page indicates this.

7 To view the certificate chain, click the plus sign (+) in front of the certificate's nickname to expand the view.

8 To view additional information about a Trusted Root certificate, click the certificate's nickname to view the Details page.

The Details page has information contained in the trusted root certificate.

9 Click **Close** > **Cancel**.

Backing Up a Server Certificate Object

OpenText Certificate Server allows you to store certificates signed by third-party certificate authorities in server certificate objects. Often these certificates cost a significant amount of money. Unfortunately, if an unrecoverable failure happens on the server that owns the certificates, the server certificate object can no longer be used. In order to protect against such failures, you might want to back up server certificates signed by external CAs and their associated private keys. Then, if a failure should occur, you can use the backup file to restore your server certificate object to any server in the tree.

The back up file contains the server's private key, public key certificate, trusted root certificate, and any intermediate CA certificates stored. This information is stored in PKCS #12 format (also known as PFX).

A server certificate object should be backed up when it is working properly.

To backup a Server Certificate object:

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificate Management**.

4 Click either the Trusted Root certificate or the public key certificate. Both certificates are written to the file during the backup operation.

5 Select the checkbox of the Server Certificate object you want to back up. > click **Export** 

This opens a wizard that helps you export the certificates to a file.

6 Select the checkbox whether to export the private key.

7 Specify a password with 6 or more alphanumeric characters to use in encrypting the PFX file. > retype password.

8 Select **Export format** > click **OK**.

9 Click **Save the exported certificate file**. Select the filename and the location for the backup file.

10 Click **Close**.

The encrypted backup file is written to the location specified. It is now ready to be stored in a secure location for emergency use.

IMPORTANT: The exported file should be put on backup media and stored in a secure place. The password used to encrypt the file should be committed to memory or stored in a vault to ensure that it is available when needed, but inaccessible to others.

Restoring a Server Certificate Object

If the Server Certificate object has been deleted or corrupted, or if the server that owned the Server Certificate object has suffered an unrecoverable failure, the object can be restored to full operation using a backup file created as described in "Backing Up a Server Certificate Object" on page 666.

If you were unable to make a backup of the server certificate object, the server certificate object might still be usable if NICI 2.x is installed on the server and a backup was made of the NICI configuration information. For information on how to back up and restore the NICI configuration files, see the "Backing Up and Restoring NICI" (https://www.netiq.com/documentation/nici27x/nici_admin_guide/data/bwf6d4c.html) section in the *NICI Administration Guide*.

To restore the Server Certificate object:

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 Delete the old server certificate object.

4 On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificate Management**.

This opens the Create a Server Certificate Wizard that creates the object.

5 In the wizard, specify the server that should own the server certificate object, and specify the certificate nickname of the server certificate. The server must have Certificate Server version 2.21 or higher installed and be up and running.

6 Select the **Import** option, then click **Next**.

7 Browse for and select the backup file, enter the backup file password, then click **Next**.

The server's private key and certificates have now been restored and the Server Certificate object is fully functional. The backup file can be stored again for future use if desired.

IMPORTANT: Be sure to protect your backup media.

Server Certificate Objects and Clustering

You can set up Server Certificate objects in a clustered environment to ensure that your cryptography-enabled applications that use Server Certificate objects always have access to them. Using the backup and restore feature for Server Certificate objects, you can duplicate the object's keying material from one node in the cluster to all nodes. Using this process for keying material signed by an external CA saves you money by allowing you to duplicate the keying material for one server certificate rather than requiring new keying material for every node in the cluster.

To set up server certificates to work in a clustered environment:

1. Create a server certificate on a server in the cluster, using either the Organizational CA or an external CA of your choice. See "Creating a Server Certificate Object" on page 647.

   When you create the server certificate objects, the Common Name (CN) portion of the certificate's subject name should be an IP or DNS name that is specific to the service. Otherwise, you receive a browser warning message indicating that the IP or DNS name on the URL does not match that in the certificate.

   If different services have different IP or DNS addresses, you need to create a server certificate for each service.

2. Back up the keying material for this server certificate object and restore it by creating a Server Certificate object with the same key pair name as the one you created in Step 1 on all remaining servers in the cluster.

   See "Backing Up a Server Certificate Object" on page 666.

Validating a Server Certificate

If you suspect a problem with a certificate or think that it might no longer be valid, you can easily validate the certificate by using OpenText Identity Console. Any certificate in the OpenText eDirectory tree can be validated, including certificates issued by external CAs.

The certificate validation process includes several checks of the data in the certificate as well as the data in the certificate chain. A certificate chain is composed of a root CA certificate and, optionally, the certificates of one or more intermediate CAs.

A result of Valid means that all certificates in the certificate chain were found to be valid. Certificates are considered valid if they pass a predefined set of criteria including whether the current time is within the validity period of the certificate, whether it has not been revoked, and whether it has been signed by a CA that is trusted. Only those certificates with a CRL distribution point extension or an OCSP AIA extension are checked for revocation.

A result of Invalid means that one or more certificates in the certificate chain were found to be invalid or their validity could not be determined. Additional information is provided for these certificates, indicating which certificate is considered invalid and why. Click Help for more information about the reason.

To validate a certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **Server Certificate Management**.

4 Select the Server Certificate object you want to validate.

5 Click **Validate Server Certificate** .

The status of the certificate is provided in the **Certificate Status** field. If the certificate is not valid, the reason is given.

Revoking a Trusted Root or Self Signed Certificate

You might find it necessary to revoke a certificate if the key or the CA becomes compromised, if the certificate has been superseded by another certificate, if the certificate is removed from the CRL, cessation of operation, and so on.

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 go to **Certificate Management** tile, click **Server Certificate Management**.

4 Select the certificate, then click  **Revoke Server Certificate**.

This starts the Revoke Certificate page. Follow the prompts to revoke the certificate.

5 Click **OK** to confirm revoking the certificate.

Moving a Server Certificate Object to a Different Server

You can move a Server Certificate object from one server to another by using the backup and restore procedures outlined in "Backing Up a Server Certificate Object" on page 666 and "Restoring a Server Certificate Object" on page 667.

1 Make sure the Server Certificate object is functional.

2 Back up the Server Certificate object.

3 Restore the Server Certificate object to the desired server.

IMPORTANT: Be sure to protect your backup media.

Replacing a Server Certificate Object's Keying Material

The private key and certificates in the server certificate object can be replaced. They should only be replaced using an internally generated PFX file created during a backup of a server certificate object. Externally generated PFX files can also be used if they contain the private key, the server certificate, and the entire certificate chain. The key and certificates in the file need not match the ones in the object; the data in the file overwrites the key and certificates in the object.

Replacing the private key and certificates in the server certificate object is a serious matter. If the key and certificates do not exactly match the ones in the object, it is the same as deleting the current server certificate object and creating a new one. See the section "Backing Up a Server Certificate Object" on page 666 for more information on the consequences of deleting the object.

If the key and certificates do match the ones in the object, replacing the keying material has no effect except to regenerate a few attributes used by the Secure Authentication Services (SAS).

To replace the keying material on the Server Certificate object:

1. As a precaution, back up the server certificate object with the private key. See "Backing Up a Server Certificate Object" on page 666.
2. Launch OpenText Identity Console.
3. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
4. go to **Certificate Management** and click **Server Certificate Management**.
5. Select the Server Certificate object you want to modify.
6. Click  **Replace Server Certificate**.

   This opens Replace Server Certificate page.
7. Choose the **File name** and enter **Password**.
8. Click **OK**.

   The operation can be started from either page. It replaces both certificates as well as the private key and any other certificates in the certificate chain.

The server's private key and certificates have now been replaced and the server certificate is fully functional. The backup file should be stored again for future use if desired.

IMPORTANT: Be sure to protect your backup media.

User Certificate Tasks

- "Creating User Certificates" on page 671
- "Creating User Certificates in Bulk" on page 671
- "Importing a Public Key Certificate into a User Object (with or without the Private Key)" on page 671
- "Viewing a User Certificate's Properties" on page 672
- "Exporting a User Certificate" on page 672

- "Exporting a User Certificate and Private Key" on page 673
- "Validating a User Certificate" on page 674
- "Revoking a User Certificate" on page 675
- "Deleting a User Certificate and Private Key" on page 675

Creating User Certificates

This task is described in "Creating a User Certificate" on page 649.

Creating User Certificates in Bulk

This feature allows you to create user certificates for multiple users at the same time, using one sequence of operations.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificate Management**.

   This opens a wizard that helps you create the user certificate.
4. Browse for and select all users you want to create a user certificate for.
5. Follow the wizard prompts to create the certificate for each user. For specific information on the wizard pages, click .

Importing a Public Key Certificate into a User Object (with or without the Private Key)

You can import any public key certificate into a user object (for example, a certificate signed by a third-party certificate authority). This certificate can appear as one of two types of files:

- **DER:** Contains a public key certificate only.
- **PFX or PKCS#12:** Contains a public key certificate as well as a private key.

After it is imported, the certificate is stored in the User object and appears on the list of certificates available.

NOTE: When importing a PKCS#12 certificate, only the public key certificate and private key are stored on the User object. No other certificates are stored. Other certificates in the user's certificate chain should probably be stored in the CN=Trusted Roots.CN=Security container (create a new Trusted Root object for each certificate in the chain).

To import a Public Key Certificate into a User object:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificate Management**.

4 Browse for and select a User object to import the public key certificate into.

5 Click **Create User Certificate** +.

6 Specify a nickname for the user certificate.

The nickname should be unique and should help you identify the certificate. You can enter up to 64 characters in the **Certificate Nickname** field.

7 Select the import creation method, then click **Next**.

8 Browse for and select the certificate to import, then click **OK**.

9 (Conditional) If you are importing a certificate with a private key, enter the password for the private key, then click **Next**.

10 Click **Finish**.

This stores the certificate in the User object, and the certificate appears on the list of certificates available to this user.

Viewing a User Certificate's Properties

In addition to the OpenText eDirectory rights and properties that are viewable with any OpenText eDirectory object, you can also view properties specific to the user certificate, including the issuer, the certificate status, the private key status, and the validation period.

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificates Management**.

4 Click the nickname of the certificate to view its details.

5 Click **Close** when you are done viewing.

Exporting a User Certificate

In order to exchange secure e-mail with another person, you must first have the other person's public key certificate. One way of obtaining that certificate is to export it using OpenText Identity Console. The other person's certificate can also be obtained by using LDAP or e-mail.

To export your own or any other user's public key certificate:

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 go to **Certificate Management** tile and click **User Certificate Management**.

4 Select the certificate, then click **Export**.

   This opens a page that helps you export the user certificate to a file. If you are logged in as the user that owns the certificate. See "Exporting a User Certificate and Private Key" on page 673.

5 If you want to export the private key then select **Export private key** checkbox.

6 Provide password > retype password.

7 Select an export format if you are not exporting the private key, then click **Next**.

8 Click **Save the exported certificate to a file** and save the file to a location of your choice.

9 Click **OK**.

Exporting a User Certificate and Private Key

In order to use a certificate for secure e-mail, authentication, or encryption, both the private key and the certificate must be available to the cryptography-enabled application. You must export the user certificate and private key and place it in a location that the application has access to in order for the application to use them.

The private keys in a user's object belong to that user. Only someone logged in as that user can export the private key. No other user, not even the network administrator, has rights to export another user's private key.

To export your own private key and certificate:

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as the user who owns the certificate.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificates**.

4 Select the certificate, then click **Export**.

   This opens a wizard that helps you export the user certificate to a file.

5 Select **Export private key** checkbox.

6 Provide password > retype password.

7 Click **Close**.

   The encrypted file is written to the location specified. It is now ready to be imported into a cryptography-enabled application.

IMPORTANT: The exported file can be kept to provide a backup. If so, it should be stored in a secure place. The password used to encrypt the file should be committed to memory or stored in a safe place to ensure that it is available when needed, but inaccessible to others.

Validating a User Certificate

If you suspect a problem with a certificate or think that it might no longer be valid, you can easily validate the certificate by using OpenText Identity Console. Any certificate in the OpenText eDirectory tree can be validated, including certificates issued by external CAs.

The certificate validation process includes several checks of the data in the certificate as well as the data in the certificate chain. A certificate chain is composed of a root CA certificate and, optionally, the certificates of one or more intermediate CAs.

A result of Valid means that all certificates in the certificate chain were found to be valid. Certificates are considered valid if they pass a predefined set of criteria including whether the current time is within the validity period of the certificate, whether it has not been revoked, and whether it has been signed by a CA that is trusted. Only those certificates with a CRL distribution point extension or an OCSP AIA extension are checked for revocation.

A result of Invalid means that one or more certificates in the certificate chain were found to be invalid or their validity could not be determined. Additional information is provided in these cases about which certificate is considered invalid and why. Click Help for more information about the reason.

To validate a certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see “Entry Rights Needed to Perform Tasks” on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificates**.
4. Select the user certificate you want to validate.
5. Click **Validate** .

   The status of the certificate is provided in the **Certificate Status** field. If the certificate is not valid, the reason is given.

NOTE: If the user certificate was signed by a third-party CA, the certificate chain must be in the Trusted Roots container in the Security container (CN=Trusted Roots.CN=Security) for the validation to succeed. Typically, the certificate chain consists of a single, root-level CA or it consists of an Intermediate CA and a root-level CA. The name of the Trusted Roots container must be Trusted Roots and each certificate in the chain must be stored in its own Trusted Root object. For instructions on how to create a Trusted Roots container and Trusted Root objects, see “Creating a Trusted Root Container” on page 649 and “Creating a Trusted Root Object” on page 650.

When validating user certificates or intermediate CA certificates signed by external CAs, the external CA’s certificate must be stored in a Trusted Root object in order for the certificate validation to be successful. The Trusted Root object must be in a Trusted Root Container named Trusted Roots and it must be located in the Security container.

Revoking a User Certificate

You might find it necessary to revoke a certificate if the key or the CA becomes compromised, if the certificate has been superseded by another certificate, if the certificate is removed from the CRL, cessation of operation, etc.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificates**.
4. Select the user certificate you want to revoke.
5. Click **Revoke** .

   This starts the Revoke Certificate Wizard. Follow the prompts to revoke the certificate.
6. Click **OK**.

Deleting a User Certificate and Private Key

If a user certificate has become invalid or you suspect the private key has been compromised in some way, you might need to delete the user certificate and private key.

Before you delete a user certificate and private key, you should revoke the user certificate. See "Revoking a User Certificate" on page 675.

To delete a user certificate and private key:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as the user who owns the certificate or as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **User Certificates**.
4. Browse for and select a User object whose certificate you want to delete.
5. Select the user certificate you want to delete.
6. Click **Delete** .

X.509 Certificate Self-Provisioning

This section describes the X.509 self-provisioning feature.

- "Overview" on page 676
- "User Self-Provisioning" on page 676
- "Server Self-Provisioning" on page 677
- "Certificate Self-Provisioning and the Issue Certificate Task" on page 678

Overview

When you create an X.509 certificate, there are many important pieces of information that must be identified and substantiated before the certificate authority (CA) issues the certificate. Two of the most important tasks are:

- Verifying the identity of the certificate's subject (verifying the identity of the person or object the certificate is for).
- Verifying the appropriateness of the subject name in the certificate (verifying that the subject name correctly represents the identity of the person or object the certificate is for).

These two tasks can be very time-consuming and are often performed by a separate administrative person or group.

OpenText Certificate Server has always leveraged the secure identity management capabilities of OpenText eDirectory to reduce the time and effort needed to perform these verifications. OpenText Identity Console allows an administrator to create user certificates in bulk; that is, to create a certificate for a large number of users at one time. The CA checks that the identity of the certificate is tied to the OpenText eDirectory account, which verifies the identity of the certificate's subject; however, the CA has not verified the appropriateness of the subject name in the certificate. Because of this, creating certificates with OpenText Certificate Server has always required that the person or software have administrative rights to the Organizational CA.

Self-provisioning allows a user or server to generate certificates without having administrative rights to the Organizational CA and without intervention of a separate administrative person or group, and still maintain the security of the CA.

OpenText Certificate Server verifies the identity of the certificate's subject by checking that the OpenText identity of the certificate is tied to the OpenText eDirectory account. The CA also verifies the appropriateness of the subject name in the certificate by checking against information in OpenText eDirectory. This allows the Organizational CA to leverage the security identity management capabilities of OpenText eDirectory to reduce administrative tasks while maintaining the security of the CA.

User Self-Provisioning

In the past, creating a user certificate required administrative rights to the CA as well as rights to attributes on the User object. With user self-provisioning, administrative rights to the CA are not necessary; however, Read (R) and Write (W) rights to the userCertificate, NDSPKI:UserCertificateInfo, and SAS:SecretStore attributes are still necessary.

If the person requesting the creation of the certificate has administrative rights to the CA, the certificate creation is not affected by whether or not user self-provisioning is enabled. If the person requesting the creation of the certificate does not have administrative rights to the CA, the subject name in the request is compared to the user's OpenText eDirectory DN and any values in the `sasAllowableSubjectNames` attribute.

If the subject name matches, the CA checks to ensure that any Subject Alternative Names are appropriate. The CA does this by checking that there is not more than one Subject Alternative Name. If the name exists, it must be of type email name and it must match a configured email name on the User object. If all these checks succeed, the CA does not require administrative rights to the CA in order to create the certificate.

To use user self-provisioning:

1 Enable user self-provisioning

   1a Launch OpenText Identity Console.

   1b Log in to the OpenText eDirectory tree as an administrator with administrative rights to the Organizational CA.

   1c On the home page, click **Certificate Management** tile > **CA Management** > click **CA Configuration** .

   1d Select **Enable user self-provisioning**.

   1e Click **Apply** > **OK**.

2 Log in to OpenText Identity Console as an administrator.

3 On the **Rights Management** tile, click **Modify Trustees** > **Search object** .

4 Browse for and select the object you want the rights to be inherited from (for example, the root of the tree or a container), then click **OK**.

5 Click **Add Trustee**, select the object, then click **OK**.

6 Click **Assigned Rights**.

7 Click **Add Property**. SELECT PROPERTY page appears.

8 Select the checkbox **Show all properties in schema**.

9 Select the `userCertificate` attribute, then click **OK**.

10 Select **Read and Write** rights.

11 Select the **Inherit** checkbox.

12 Repeat Step 5 through Step 11 for the other attributes (`NDSPKI:UserCertificateInfo` and `SAS:SecretStore`).

13 Click **Done** > **Apply**.

14 Click **OK**.

Server Self-Provisioning

In past, creating a server certificate required administrative rights to the CA as well as administrative rights to the context the server certificate was to be created in. With server self-provisioning, administrative rights to the CA are not necessary; however, administrative rights to the context the server certificate was created in are still necessary.

To create the server certificate, you must have the administrative rights to the CA. The certificate creation is not affected by whether or not server self-provisioning is enabled. In case you do not have the required administrative rights to the CA, enable the **Require read rights to operate the CA** option to operate the CA from the **Certificate Management** tile in OpenText Identity Console. The administrative rights to the CA are not required if any one of the following are true:

- The subject name in the request is compared to the server's OpenText eDirectory DN and any IP or DNS addresses as determined by a DNS or eDirectory SLP lookup. If the subject name matches either, the CA does not require administrative rights to the CA in order to create the certificate.

- Non CN components of the subject name matches non CN components of CA certificate's subject name.
- Subject alternative name only has IP-address/DNS name that is verified again by CA through reverse DNS lookup.

The servers are not granted write rights to the CA's **NDSPKI:Private Key** attribute by default. If **Require write rights to operate the CA** is enabled from the **Configure Certificate Authority** task in OpenText Identity Console, you should grant servers the write rights to the CA's **NDSPKI:Private Key** attribute.

NOTE: Be aware that when PKI Health Check runs on a server with server self-provisioning enabled, your server's server certificates might be automatically created (if none exist) or replaced (if they are expired). For more information, see "PKI Health Check" on page 692

To use server self-provisioning:

1. Ensure you have OpenText eDirectory 9.0 and the OpenText Identity Console application installed.

   Both eDirectory 8.8 and the OpenText Identity Console application are included with OES 2 and are installed automatically when you select any of the OpenText eDirectory-required components during the OES 2 installation.

2. Enable server self-provisioning:

   2a Launch OpenText Identity Console.

   2b Log in to the OpenText eDirectory tree as an administrator with administrative rights to the Organizational CA.

   2c go to **Certificate Management** tile > **CA Management** and click **CA Configuration** .

   2d Select **Enable user self-provisioning**.

   2e Click **Apply** > OK.

Certificate Self-Provisioning and the Issue Certificate Task

The Issue Certificate task allows the creation of a certificate by using a PKCS#10 certificate signing request (CSR). This task allows the user to create a certificate that is not tied to any OpenText eDirectory object. If the person requesting the creation of the certificate has administrative rights to the CA, the certificate creation is not affected. If the person requesting the creation of the certificate does not have administrative rights to the CA, the certificate request is treated as a user self-provisioning request, but the person does not need to have rights to the attributes userCertificate, NDSPKI:UserCertificateInfo, and SAS:SecretStore attributes on the object. This is because the certificate is not stored in OpenText eDirectory, so rights to the object are not needed.

User self-provisioning must be enabled for a user to issue certificates without having administrative rights to the CA. Complete Steps 1 through 3 of "User Self-Provisioning" on page 676.

For information on the Issue Certificate task, see "Issuing a Public Key Certificate" on page 653.

Using OpenText eDirectory Certificates with External Applications

Some customers use non-eDirectory applications that require X.509 certificates and keys (for example, Apache or OpenSSL). Most of these applications are configured out of the box with self-signed (no value) certificates, which are meant only to provide a temporary solution until the application can be configured with real X.509 certificates and keys.

Unfortunately, many administrators do not replace these self-signed certificates, often because it is too time-consuming or too difficult. In addition, X.509 certificates are designed to expire regularly, so replacing them on a regular basis is an ongoing administrative task.

The following sections describe the solution to this problem:

- "PKI Health Check Functionality" on page 679
- "Configuring the SAS:Service Object to Export eDirectory Certificates" on page 680

PKI Health Check Functionality

In response to customer requests to provide non-eDirectory applications with X.509 certificates, the PKI Health Check code within OpenText Certificate Server now provides the capability to automatically export X.509 certificates and keys to the file system, enabling non-eDirectory applications to take advantage of OpenText eDirectory-minted certificates and eOpenText Directory-managed certificates.

When the PKI Health Check runs, it automatically overwrites any existing certificates, including the certificates' private keys. However, to ensure that no valid certificates and private keys are deleted, the PKI Health Check determines whether the existing certificates and keys are the same as those configured in OpenText eDirectory. If they are different than those configured in OpenText eDirectory, the PKI Health Check creates a backup of these files before overwriting them. This ensures that certificates that have been acquired from an external source (for example, VeriSign*) are not deleted.

After a configuration has been created for the server on the SAS:Service Object, keys and certificates associated with the specified server are automatically exported to the file system. If the keys and certificates are replaced or updated in OpenText eDirectory (for example, if the Server Certificate object is deleted and a new one is created with the same name), the new keys and certificates are automatically exported to the file system the next time PKI Health Check runs.

NOTE: The PKI Health code within OpenText Certificate Server runs once every time OpenText Certificate Server loads/reloads. You can use any of the following methods to reload the OpenText Certificate Server:

- Restart the server
- Restart OpenText eDirectory
- Unload and load PKI Server manually
- Run an OpenText eDirectory repair (NDSRepair)

  OpenText Certificate Server shuts down during the repair and reloads after the OpenText eDirectory repair is finished.

For more information on the PKI Health Check, see "PKI Health Check" on page 692.

Before the PKI Health Check can automatically export X.509 certificates and keys to the file system, the SAS:Service Object must be configured. This is because the PKI Health Check reads the configuration on the SAS:Service Object. For information on how to configure the SAS:Service Object, see "Configuring the SAS:Service Object to Export eDirectory Certificates" on page 680.

Configuring the SAS:Service Object to Export eDirectory Certificates

Before an OpenText eDirectory Server Certificate can be exported to the file system, a configuration must first be created for the server on the SAS:Service Object. This can be done either automatically or manually, depending on what OpenText eDirectory server you are using. The following sections further explain these options:

- "Manually Configuring the SAS:Service Object to Enable Use of eDirectory Certificates" on page 680

Manually Configuring the SAS:Service Object to Enable Use of eDirectory Certificates

If you are not using OES 2 as your OpenText eDirectory server, you must manually configure the SAS:Service Object in order to export eDirectory certificates. This configuration must specify the Server Certificate name. If multiple server certificates need to be exported, you can simply create multiple configurations. You can export the same certificate to a different file path, or you can export a different certificate to a different file path.

NOTE: Each configuration must use unique file paths in order to avoid file collisions. The Public key path and the Private key path must be unique and different from each other and from any other configuration.

To create a configuration on the SAS:Service object:

1. On the OpenText Identity Console home page, click **Certificate Management** tile > **SAS Service Objects**.
2. Click **SAS Service Object**.
3. Select the SAS:Service object where you want to create the configuration.
4. Click **Create New** + to create Synchronized server certificate. The Server Certificate Synchronization window is displayed.
5. In the **Certificate** field, browse  for the certificate you want to export.
6. In the **Public key path** field, specify the path where the application will find and use the certificate. For example: `C:/novell/nds/servercert.pem`.
7. In the **Private key path** field, specify the path where the application will find and use the certificate's private key. For example: `C:/novell/nds/serverkey.pem`.
8. Select the key type that you are going to use. If you are running OpenSSL, select PKCS#8. If you are running Apache, select PKCS#1.
9. Click **OK**.

   The configuration is created. The name, path, key path and key type are displayed.
10. Click **Save**.

To create another configuration, repeat Step 4 through Step 9.

If you are using a OES server as your OpenText eDirectory server, then you can automatically configure the server to create a configuration on the SAS:Service Object.

NOTE: If use of OpenText eDirectory certificates is enabled while installing OES 2 (default), the install code creates a configuration for the SSL CertificateDNS object, and the certificates and keys are exported to the following files:

key file - `/etc/ssl/servercerts/serverkey.pem`

certificate file - `/etc/ssl/servercerts/servercert.pem`

Trusted Root Object Tasks

- "Creating a Trusted Root Container" on page 681
- "Creating a Trusted Root Object" on page 681
- "Viewing a Trusted Root Object's Properties" on page 681
- "Validating a Trusted Root Object" on page 682
- "Revoking a Trusted Root Certificate" on page 682

Creating a Trusted Root Container

This task is described in "Creating a Trusted Root Container" on page 649.

Creating a Trusted Root Object

This task is described in "Creating a Trusted Root Object" on page 650.

Viewing a Trusted Root Object's Properties

In addition to the OpenText eDirectory rights and properties that are viewable with any OpenText eDirectory object, you can also view properties specific to the Trusted Root object, including the issuer, the certificate status, and the validation period.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **Trusted Root Management**.
4. On the **Trusted Root** tab > select the nickname of the certificate to view its details.
5. Click **Cancel** to close the page.

Validating a Trusted Root Object

If you suspect a problem with a certificate or think that it might no longer be valid, you can easily validate the certificate by using OpenText Identity Console. Any certificate in the OpenText eDirectory tree can be validated, including certificates issued by external CAs.

The certificate validation process includes several checks of the data in the certificate as well as the data in the certificate chain. A certificate chain is composed of a root CA certificate and, optionally, the certificates of one or more intermediate CAs.

A result of Valid means that all certificates in the certificate chain were found to be valid. Certificates are considered valid if they pass a predefined set of criteria including whether the current time is within the validity period of the certificate, whether it has not been revoked, and whether it has been signed by a CA that is trusted. Only those certificates with a CRL distribution point extension or an OCSP AIA extension are checked for revocation.

A result of Invalid means that one or more certificates in the certificate chain were found to be invalid or their validity could not be determined. Additional information is provided in these cases about which certificate is considered invalid and why. Click **Help** for more information about the reason.

To validate a Trusted Root certificate:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **Trusted Root Management**.
4. On the **Trusted Root** tab > select the certificate, then click **Validate Trusted Root Certificate**.

   The status of the certificate is provided in the **Certificate Status field**. If the certificate is not valid, the reason is given.

NOTE: If the certificate in the object is not self-signed, its certificate chain must be in the Trusted Roots container in the Security container (CN=Trusted Roots.CN=Security) for the validation to succeed. Typically, the certificate chain consists of a single, root-level CA or it consists of an Intermediate CA and a root-level CA. The name of the Trusted Roots container must be Trusted Roots and each certificate in the chain must be stored in its own Trusted Root object. For instructions on how to create a Trusted Roots container and Trusted Root objects, see "Creating a Trusted Root Container" on page 649 and "Creating a Trusted Root Object" on page 650.

Revoking a Trusted Root Certificate

You might find it necessary to revoke a certificate if the key or the CA becomes compromised, if the certificate has been superseded by another certificate, if the certificate is removed from the CRL, cessation of operation, etc.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **Trusted Root Management**.

4 Browse to and click the Trusted Root object you want to modify.

5 Select the certificate, then click **Revoke**.

This starts the Revoke Certificate Wizard. Follow the prompts to revoke the certificate.

6 Click **OK > Finish**.

Certificate Revocation List (CRL) Tasks

OpenText Certificate Server provides a system for managing Certificate Revocation Lists (CRLs). This is an optional system, but it must be implemented if you want to be able to revoke certificates created by the Organizational CA.

A CRL is a published list of revoked certificates and the reason the certificates were revoked.

- "Creating a CRL Container Manually" on page 683
- "Deleting a CRL Container" on page 684
- "Creating a CRL Configuration Object" on page 684
- "Activating a CRL Configuration Object" on page 685
- "Viewing and Modifying a CRL Configuration Object's Properties" on page 685
- "Deleting a CRL Configuration Object" on page 687
- "Creating a CRL Object" on page 687
- "Exporting a CRL File" on page 688
- "Replacing a CRL File" on page 688
- "Extending Validity of CRL File" on page 689
- "Viewing a CRL Object's Properties" on page 689
- "Deleting a CRL Object" on page 690

Creating a CRL Container Manually

During the Certificate Server installation, a CRL container is created if the user has the appropriate rights to create it. If not, the CRL container can be created manually by someone with the appropriate rights after the installation is completed.

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **CA Management** > **CRL** tab.

If a CRL container already exists, you are brought to the Organizational CA's property page.

If no CRL container exists, this launches a wizard that creates a CRL container and a CRL Configuration object to go in the container.

4 Follow the wizard to completion.

NOTE: If the CRL container is created in a different container other than the Security container, the `ndspkiCRLContainerDN` attribute has to be populated manually on the tree CA object for the CRL tab to list the CRLs.

Deleting a CRL Container

Deleting a CRL container is possible, but it is not recommended.

The general rule is to not delete a CRL container, CRL configuration object, CRL object, or CRL file until one issue date after the last certificate that contains a related distribution point has expired.

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 On the OpenText Identity Console home page, click **Certificate Management** tile > **CA Management** > **CRL** tab.

4 Browse for and select the CRL container you want to delete.

5 Click **Delete** > **OK**.

Creating a CRL Configuration Object

A CRL Configuration object can be created in the CRL container. This is an object that contains the configuration information for the CRL objects that are available in the OpenText eDirectory tree. Normally, you have only one CRL Configuration object in your tree. You might need multiple CRL Configuration objects if you are creating or rolling over a new Organizational CA, but only one CRL Configuration object can be used to create new certificates.

The CRL Configuration object resides in the CRL container.

1 Launch OpenText Identity Console.

2 Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.

3 Go to **Certificate Management** tile > **CA Management** > **CRL** tab and then perform one of the following:

- If no CRL container exists, then click + to create a CRL container. Specify all the details in the page that appears.
- If a CRL container exists, but no CRL Configuration object exists, select the CRL Container to modify the details in CRL Configuration and CRL information tabs.

4 Click **Save**.

5 Click **OK**.

NOTE: Ensure that the CRL file path which is specified here, is in respect with the OpenText eDirectory installation path.

Activating a CRL Configuration Object

Only one CRL Configuration object can be active in an OpenText eDirectory tree at one time. If you have more than one CRL Configuration object, you must choose which one to activate. By default, the first CRL Configuration object created is active.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **CA Management** > **CRL** tab.
4. Select a CRL Configuration object, then click **Make Active** .

Viewing and Modifying a CRL Configuration Object's Properties

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page, click **Certificate Management** tile > **CA Management**.
4. Click the **CRL** tab.
5. Click on the name of the CRL Configuration object you want to view or modify.
6. Click **OK** or **Apply**.

NOTE: The CRL configuration can be disabled while validating the certificates. To disable the CRL configuration, you must set the environment variable `NDSD_DISABLE_CRL_CONFIG` to any value. If your OpenText eDirectory tree is already configured with CRL, ensure to remove the CRL configuration objects (`objectclass: ndspkiCRLConfiguration`) and CRL Distribution point objects (`objectclass: cRLDistributionPoint`) manually before upgrading OpenText eDirectory.

- "LDAP Mapping" on page 686
- "HTTP Distribution Point Location" on page 686

LDAP Mapping

The standard LDAP type for Certificate Revocation Lists limits the size of the CRL to 64 KB. To change this limitation, you must create the CRL directory entries with OpenText-defined types. In order for the LDAP distribution points to be found, you must map the standard LDAP types to the OpenText LDAP types by doing the following:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory as an administrator with the appropriate rights.
3. On the OpenText Identity Console home page, click **LDAP Configuration** tile.
4. On the **Type** drop down, select **View LDAP Groups** > **Search**, then select the LDAP group that needs to be mapped.
5. On the Attribute Map page, and make the following changes:
   - **5a** The default mapping from Primary LDAP Attribute certificateRevocationList; binary (and secondary attribute certificateRevocationList) to the OpenText eDirectory attribute certificateAuthorityList should be changed to the OpenText eDirectory attribute ndspkiCertificateRevocationList (that is, change the OpenText eDirectory attribute from certificateAuthorityList to ndspkiCertificateRevocationList).
   - **5b** The default mapping from Primary LDAP Attribute authorityRevocationList;binary (secondary attribute authorityRevocationList) to the OpenText eDirectory attribute authorityRevocationList should be changed to the OpenText eDirectory attribute ndspkiAuthorityRevocationList (that is, change the OpenText eDirectory attribute from authorityRevocationList to ndspkiAuthorityRevocationList).
   - **5c** The default mapping from Primary LDAP Attribute deltaRevocationList;binary (secondary attribute deltaRevocationList) to the OpenText eDirectory attribute deltaRevocationList should be changed to the OpenText eDirectory attribute ndspkiDeltaRevocationList (i.e. change the OpenText eDirectory attribute from deltaRevocationList to ndspkiDeltaRevocationList).
6. Click **Save**.
7. On the **Type** drop down, select **View LDAP Servers** > **Search**.
8. Select the server that hosts the LDAP distribution point.
9. Click the **Information** drop down.

   This restarts the LDAP service, and it begins using the correct mapping for the CRL attributes.

   For more information on LDAP management, see Chapter 14, “Configuring LDAP Services for OpenText eDirectory,” on page 365.

HTTP Distribution Point Location

When configuring Certificate Server to use an HTTP distribution point, it is important that you specify a location that is accessible to users wanting to validate certificates. If a user cannot locate a CRL for a certificate containing a distribution point, the certificate is considered invalid. The distribution point must be located in a directory that is available to the Web server specified by the HTTP address in the distribution point. If that directory is not on the same server that is hosting the Certificate Authority, the CRL must be moved manually, with a script, or created on a mounted directory.

Deleting a CRL Configuration Object

Deleting a CRL Configuration object is possible, but it is not recommended. When a CRL Configuration object is deleted, the server quits creating the CRL files. If a CRL file already exists in the location specified in the CRL object, certificate validation continues to use it until it expires. After it expires, all certificates that have a CRL distribution point that references that CRL file fail validation.

The general rule is to not delete a CRL container, CRL configuration object, CRL object, or CRL file until one issue date after the last certificate that contains a related distribution point has expired.

1. Launch OpenText Identity Console
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702
3. On the OpenText Identity Console home page > **CA Management**.
4. Select the CRL Configuration object you want to delete.
5. Click **Delete** .

Creating a CRL Object

This task allows you to create a CRL object (cRLDistributionPoint) to store third-party CRLs in OpenText eDirectory. This object can be created in any container in the OpenText eDirectory tree. But as a general rule, OpenText CRL objects reside in a CRL Configuration object and do not need to be created manually. A CRL object is automatically created for you when you create a CRL Configuration object.

The CRL object contains a CRL file, which contains the detailed CRL information. For a OpenText CRL object, the CRL file is automatically created and updated whenever the server issues a new one. For other CRL objects, you must import a CRL file from a third-party CA.

NOTE: The term CRL Distribution Point is used in different ways. It is the OpenText eDirectory schema object name for the CRL object and it can be used in general terms as the point where the CRL information is published.

To create a CRL object:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **CA Management** > **Create CRL**.
4. Type a name for the object and provide the context where you want the object to reside.
5. Paste a copy of the CRL into the field or read it from a CRL file.
6. Click **Nest** to create the object.

Exporting a CRL File

You can export the CRL that is contained in the CRL Distribution Point object to a file.

To export a OpenText CRL file:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **CA Management** > **Create CRL**.
4. Click the **CRL** tab.
5. Click the name of the CRL Configuration object.
6. Click **Export CRL**.
7. Select an output format, then click **Next**.
8. To save the exported CRL to a file, click **Save**, then specify a location for the file.
9. Click **OK**.

To export a third-party CRL file:

1. On the OpenText Identity Console home page, click **Certificate Management** tile.
2. Click **CA Management** > **CRL** tab > select the check box of the CRL that you want to export.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. Click **Export CRL**.
4. Select an output format, then click **Next**.
5. To save the exported CRL to a file, click **Save**, then specify a location for the file.
6. Click **OK** > **OK**.

Replacing a CRL File

You can replace a CRL file, but it is not recommended.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** > **CA Management**.
4. Click the **CRL** tab.
5. Click the name of the CRL Configuration object, then click **Details**.
6. Click **Replace** .
7. Click **OK** to continue.
8. Browse for and select the new CRL file.
9. Click **OK**.

If a CRL file does not exist on the CRL Configuration object, the **Import** button is displayed.

Extending Validity of CRL File

The administrator can extend the validity of the CRL file using Identity Console. To extend the validity, perform the following steps:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > Certificate Management > **CA Management**.
4. Click the **CRL** tab.
5. Click the name of the CRL file.
6. Select **Extend validity by following hour(s)** under **Next CRL Issuance** and mention the number of hours in the next box. You can enter any value ranging from 1 to 12 hours in this field.
7. Click **Issue Now** > click **Save**.

Viewing a CRL Object's Properties

To view a OpenText CRL object's properties:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** > **CA Management**.
4. Click the **CRL** tab.
5. Click the name of the CRL Configuration object.

   You can now view the CRL object's properties.
6. When you are finished viewing properties, click **Save** or **Cancel**.

To view a third-party CRL object's properties:

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > **Certificate Management** > **CA Management**.
4. Click the **CRL** tab.

   You can now view the CRL object's properties.
5. When you are finished viewing properties, click **OK** or **Apply**.

Deleting a CRL Object

If you delete a CRL object, it is re-created the next time the server generates the CRL file. If you delete a CRL object that you created using Identity Console and import it, then it is gone permanently and any certificates that reference it are considered invalid.

The general rule is to not delete a CRL container, CRL configuration object, CRL object, or CRL file until one issue date after the last certificate that contains a related distribution point has expired.

1. Launch OpenText Identity Console.
2. Log in to the OpenText eDirectory tree as an administrator with the appropriate rights.

   To view the appropriate rights for this task, see "Entry Rights Needed to Perform Tasks" on page 702.
3. On the OpenText Identity Console home page > Certificate Management > **CA Management**.
4. Browse to and click the CRL object you want to delete.
5. Click **Delete** > **OK**.

OpenText eDirectory Tasks

- "Resolving Multiple Security Containers, Organizational CAs, KAP Containers, and W0 Objects" on page 690
- "Restoring or Re-creating a Security Container" on page 691
- "Restoring or Re-creating KAP and W0" on page 691

Resolving Multiple Security Containers, Organizational CAs, KAP Containers, and W0 Objects

OpenText Certificate Server can be installed on multiple servers in an OpenText eDirectory tree. However, for OpenText Certificate Server to function properly, only one Security container, Organizational CA, KAP container, and W0 object should exist in the tree.

If you are installing OpenText Certificate Server on multiple servers in an OpenText eDirectory tree, you must allow OpenText eDirectory to replicate between each installation of OpenText Certificate Server. If you do not allow OpenText eDirectory to replicate, your installation to another server might not recognize that the tree already has a Security container, an Organizational CA, a KAP container, and W0 object and might re-create these objects on another server in the same eDirectory tree.

The items below describe possible scenarios and how to resolve them.

- If you have two or more Security containers in the same OpenText eDirectory tree and each contains an Organizational CA, and a KAP container with a W0 object, do not issue any certificates. Contact Technical Support for help in resolving this.
- If you have one Security container that contains two KAP containers in the same OpenText eDirectory tree, do not issue any certificates. Contact Technical Support for help in resolving this.

- If you have one Security container that contains two Organizational CAs and one KAP container with a W0 object in the same OpenText eDirectory tree, delete every server and user certificate issued by both Organizational CAs. Then, delete both CAs and create a new Organizational CA. Issue new server and user certificates as needed.
- If you have two or more Security containers in the same OpenText eDirectory tree and each contains an Organizational CA, but only one contains a KAP container with a W0 object, delete every server and user certificate issued by all Organizational CAs. Delete all the Security containers without the KAP container and W0 object. If the remaining Security container is not named *Security*, rename it to *Security*. Issue new server and user certificates as needed.
- If you have two or more Security containers in the same OpenText eDirectory tree and only one contains an Organizational CA and a KAP container with a W0 object, delete all the Security containers without the KAP container and W0 object. If the remaining Security container is not named *Security*, rename it to *Security*.

Restoring or Re-creating a Security Container

If you delete the Security container, you cannot create an Organizational Certificate Authority until you have restored or re-created the security container.

To restore the security container, you must restore the OpenText eDirectory partition containing the Security container.

To re-create the Security container, use one of two methods:

- Using OpenText Identity Console, click **Tree View** tile > **Add** > click **Tree's Security Container** > give a **Name** > then click **Next**. The container name must be Security.
- Reinstall OpenText Certificate Server on any server in the OpenText eDirectory tree.

Restoring or Re-creating KAP and W0

Do not delete the KAP or W0 objects. Doing so invalidates all previously created User certificates. If you delete one of these objects, refer to the TID #3032354, “How to Restore or Recreate KAP and W0 Objects,” for information on how to resolve this problem. You should not attempt further installations of OpenText Certificate Server, Single Sign-on, NMAS, or OpenText eDirectory until the problems have been corrected.

Application Tasks

This section describes how to configure cryptography-enabled applications to use OpenText certificates.

Some of the information in this section is dated but useful. For the latest information on using certificates with your cryptography-enabled applications, refer to the application's documentation.

The general process for enabling applications for secure e-mail is:

1. Export your Organizational CA's self-signed certificate (see “Exporting the Organizational CA's Self-Signed Certificate” on page 655), your user certificate, and the matching private key to a `.pfx` file (see “Exporting a User Certificate and Private Key” on page 673).

2. Import the `.pfx` file into your e-mail client.
3. Configure your e-mail client to secure your e-mail.

In order to create an SSL connection to a server on the Internet with your browser, you must trust the CA that signed the user or server's certificates. If you do not, your application might present you with an error. Some applications provide a warning with the ability to accept or reject the user or server certificate whose CA isn't yet known to the application.

Server and user certificates signed by a company's Organizational CA always generate such warnings and errors. This is because the Organizational CA is not listed as a trusted CA in your application. The warnings and errors can be prevented by installing the self-signed certificate of the Organizational CA into your application.

Installing the Organizational CA into your browser automatically adds it as a trusted CA.

To accept the Organizational CA as a trusted CA in your application:

1. Export your Organizational CA's self-signed certificate (see "Exporting the Organizational CA's Self-Signed Certificate" on page 655).

   NOTE: The Internet browsers recognize certificates in .der or a .crt format.

2. Import the certificate into your browser according to the directions provided by the browser documentation.

PKI Health Check

OpenText Certificate Server incorporates a process that maintains the health and integrity of the Certificate Server components. This process is called the PKI Health Check and it runs when:

- The server reboots.
- OpenText eDirectory comes up.
- DSRepair finishes running.

When PKI Health Check runs, it performs the following tasks:

Table 25-1 *PKI Health Check Tasks*

Task	Function
Verifies the server's link to the SAS Service object	This task checks to see if there is a link from the server object to a SAS:Service object. If the link exists, the task makes sure that the object is named correctly and is in the same context as the server. If the link does not exist, the task checks to see if a correctly named object exists in the same context as the server. If such an object exists, the task creates a link from the server to the object.

Task	Function
Verifies the SAS Service object	This task checks to see if a SAS:Service object exists. If it does not exist, the task creates one and creates a link from the server object to the new object. Then, the task checks to see if the SAS:Service object has the necessary OpenText eDirectory rights. If it does not, the task attempts to give the SAS:Service object the rights it needs.
Verifies the links to the KMOs	This task reads the list of Server Certificate objects (or KMOs) that are linked to the SAS:Service object. It checks whether the KMOs are all named correctly and attempts to fix their names if they are not. The task also checks whether the KMOs are all in the same context as the server object and attempts to move them to the correct context if they are not.
Checks the Server Certificates (KMOs)	This task reads all the names of KMOs that are in the same container as the server object and puts them in a list. The task then performs the following for each KMO in the list: ◆ Attempts to populate the NDSPKI:Not Before and NDSPKI:Not After attributes with the validity dates of the certificate. ◆ Checks whether Public has the Read right to the Host Server attribute. ◆ Checks the link from the KMO to a server that is a back link. If the back link is for a different server, it ignores the KMO and removes it from its list. ◆ Reads the private key and attempts to unwrap it.
Reverifies the links to the KMOs	This task reads the list of Server Certificate objects (or KMOs) that are linked to the SAS:Service object. It compares each KMO in this list to the list created in Checks the Server Certificates (KMOs). Using the checks from Checks the Server Certificates (KMOs), the task determines if there are any problems with the linked certificates and it unlinks them if the KMO is unusable. The task also determines if there are any unlinked KMOs that are usable by this server and it links them, if they exist.

Task	Function
Creates default certificates	This task determines if Server Self-Provisioning is enabled at the Organizational CA object. If Server Self-Provisioning is not enabled, this step is skipped. If Server Self-Provisioning is enabled, then the task calls the NPKICreateDefaultCertificates() API. This API creates or replaces the SSL CertificateDNS certificate if: ◆ The certificate does not exist. ◆ The certificate is not expired or about to expire. ◆ The certificate's subject name does not match the default IP and DNS address configured for the server. **NOTE:** OpenText eDirectory does not automatically create SSL CertificateIP. SSL Certificate DNS contains all the IPs listed in the Subject Alternative Name. In addition, this API acquires all of the IP and DNS addresses configured for the server and it creates and/or replaces a certificate for each one, such as IP AG *ip address* or IP DNS *dns name* if: ◆ The certificates do not exist. ◆ The certificates are expired or about to expire.
Synchronizes certificates for external services	This task reads the configuration from the SAS:Service object. For each configured entry, the task acquires the certificates and private key from the specified KMO object. If the specified directory does not exist, the task attempts to create it. The task then unwraps the private key and converts it to the specified raw-key format. The task compares any existing private key and certificate files to the ones from the specified KMO. If the keys and certificates are not the same, the task makes a backup of the existing private key and certificate files and then it overwrites them with the private key and certificates. The keys are written out in a PEM format.

Task	Function
Exports the eDirectory CA certificate to the file system	The way in which this task is accomplished depends on the operating system you are running. The SSCert.der and SSCert.pem files contain the RSA certificate of the Organization CA. If the Organization CA has ECDSA certificate, OpenText eDirectory exports the ECDSA certificate to the SSECCert.der and SSECCert.pem files and staores them in the same directory as the SSCert.der and SSCert.pem files. ◆ **Windows:** Checks if the SSCert.der and SSCert.pem files in the PKI working directory contain the same certificate as the Organization CA certificate in OpenText eDirectory. It attempts to replace the files if they are not the same. The default PKI working directory is c:\Novell\NDS\DIBFiles\CertServ\ ◆ **Linux (Not OES Linux):** Checks if the SSCert.der and SSCert.pem files in the OpenText eDirectory data directory contain the same certificate as the Organizational CA certificate in OpenText eDirectory. It attempts to replace the files if they are not the same. The default OpenText eDirectory data directory is /var/opt/novell/eDirectory/data ◆ **OES Linux:** Checks if the /etc/opt/novell/certs/SSCert.der and /etc/opt/novell/certs/SSCert.pem files contain the same certificate as the Organizational CA certificate in OpenText eDirectory. If the certificates are not the same, the task attempts to replace the files by adding the Organizational CA certificate to the /etc/ssl/certs directory and then running the c_rehash program. Before replacing the files, however, the task creates backups of any existing certificates.

Public Key Cryptography Basics

This sections describes the basics of Public Key Cryptography.

- “Overview” on page 696
- “Secure Transmissions” on page 696
- “Key Pairs” on page 696
- “Establishing Trust” on page 699

Overview

The content of most Internet communications, such as Web page browsing or public chat forums, can be monitored by anyone equipped to do so. The content of other data transmissions, such as the exchange of credit card information for online purchases, needs to be kept private.

Public key cryptography is a widely used method for keeping data transmissions private and secure on the Internet. Specifically, public key cryptography is the system of using digital codes called “keys” to authenticate senders of messages and to encrypt message content.

Secure Transmissions

Data transmissions are private and secure when two things happen:

- **Authentication:** The data receiver knows that the data sender is exactly who or what it claims to be.
- **Encryption:** The data sent is encrypted so that it can be read only by the intended receiver.

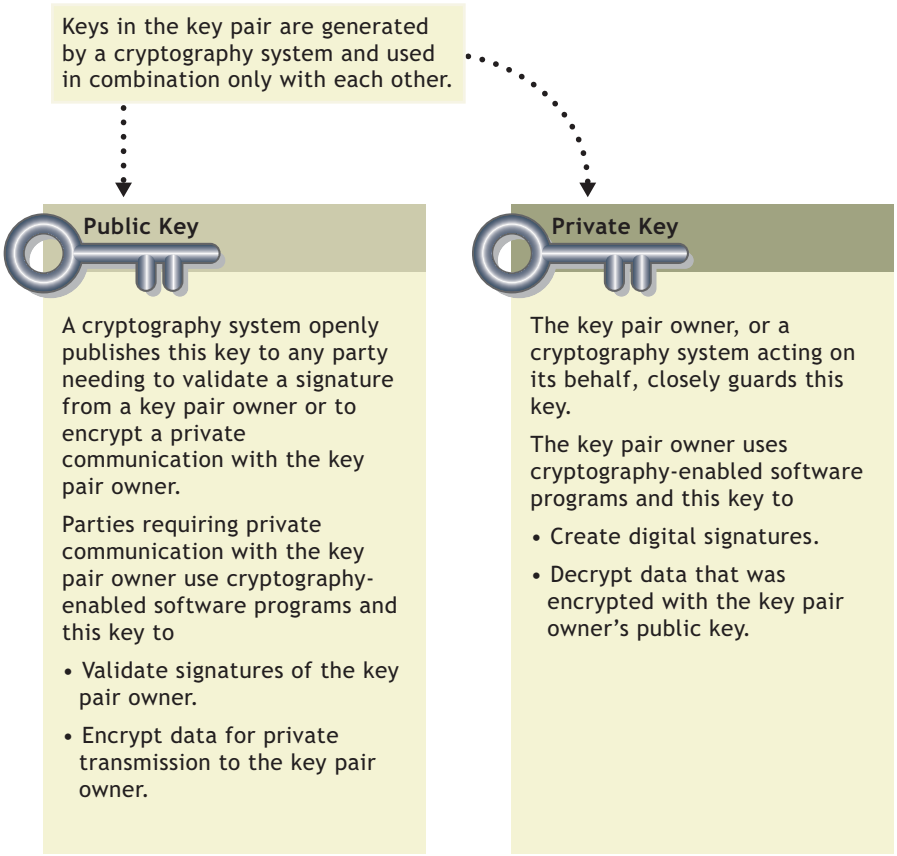
Key Pairs

Authentication and encryption are both provided through the use of mathematically related pairs of digital codes or “keys.” One key in each pair is publicly distributed; the other is kept strictly private.

Each data transmitter, whether it is a person, a software program, or some other entity such as a bank or business, is issued a key pair by a public key cryptography system.

The basic principles and functions of each key in the key pair are summarized in the following illustration:

Figure 25-1 *Basic Key Pair Description*



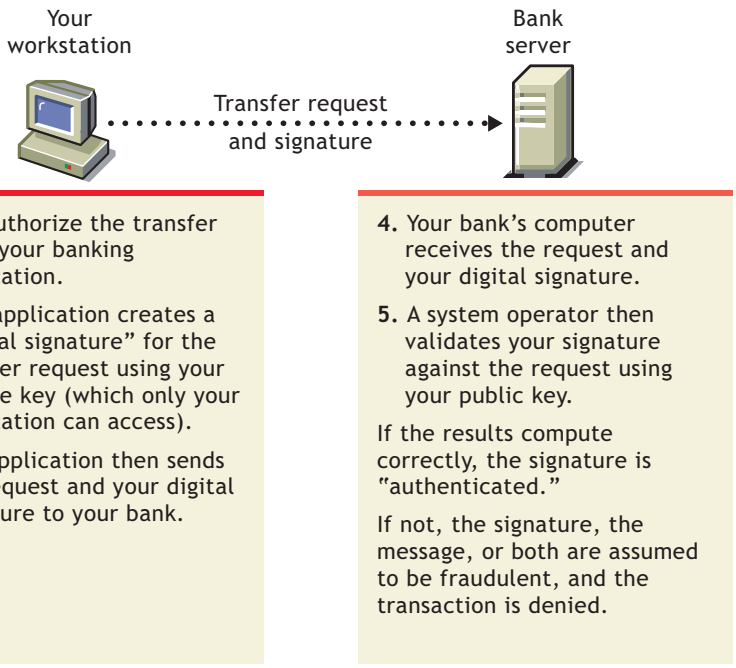
- "Key Pairs and Authentication" on page 697
- "Key Pairs and Encryption" on page 698

Key Pairs and Authentication

Authentication means that the data receiver knows that the data sender is exactly who or what it claims to be.

Suppose that you want to authorize your bank to transfer funds from your account to another account. The bank needs proof that the message came from you and that it has not been altered during transit. The following illustrates the process that your online transaction would follow, using public key cryptography.

Figure 25-2 *Public Key Process*



For information about digital signatures and their verification, see "Digital Signatures" on page 700.

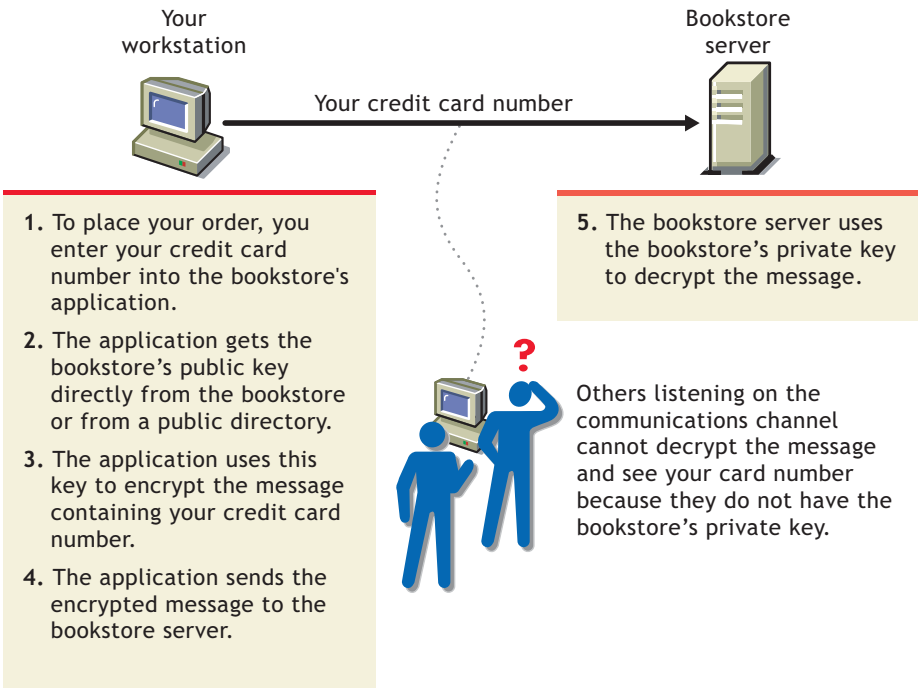
Key Pairs and Encryption

Encryption means that the data can be read only by the intended receiver.

Suppose you want to order a book from an Internet vendor and you need to use your credit card to pay for it. You don't want your credit card number read by anyone other than the intended recipient.

The encryption process in the following illustration provides the mechanisms through which your credit card number can be safely transmitted.

Figure 25-3 *Encryption Process*



Establishing Trust

If a sender and receiver know and trust each other, they can simply exchange public keys and establish secure data transmission, including authentication and encryption. To do this, they would use each other's public keys and their own private keys.

Under normal circumstances, however, parties needing secure data transmissions have no foundation for trusting the identity of each other. Each needs a third party, whom they both trust, to provide proof of their identity.

- "Certificate Authorities" on page 699
- "Digital Signatures" on page 700
- "Certificate Chain" on page 701
- "Trusted Roots" on page 702

Certificate Authorities

A party needing to prove its identity in a public key cryptography environment enlists the services of a trusted third party known as a certificate authority.

The primary purpose of the certificate authority is to verify that a party is who or what it claims to be, and then to issue a public key certificate for that party to use. The public key certificate verifies that the public key contained in the certificate belongs to the party named in the certificate.

Figure 25-4 *Certificate Request*



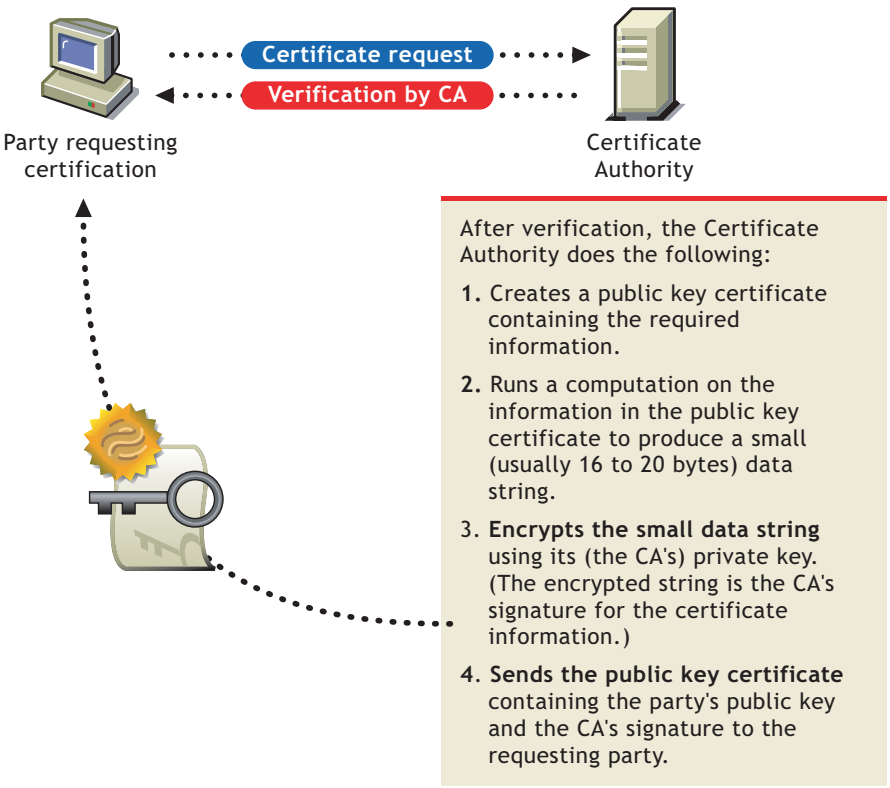
After the identity of the requesting party has been established to the satisfaction of the certificate authority, the certificate authority issues an electronic “certificate” and applies its digital signature.

Digital Signatures

Just as a personal signature applied to a paper document indicates the authenticity of the document, a digital signature indicates the authenticity of electronic data.

To create a digital signature, the software used to create the signature links the data being signed with the private key of the signer. The following illustration shows the process that a CA follows to create its digital signature for a public key certificate.

Figure 25-5 *Digital Signature*



A digital signature is uniquely linked to the signer and the data. No one else can duplicate the signature because no one else has the signer's private key. In addition, the signer cannot deny having signed the data. This is known as *non-repudiation*.

When a certificate authority signs a public key certificate, it guarantees that it has verified the identify of the public key owner according to the certificate authority's established and published policies.

After signed data (such as a public key certificate) is received, software verifies data authenticity by applying the same computation to the data that the signing software used originally. If the data is unaltered, both computations produce identical results. It can then be safely assumed that neither the data nor the signature was modified in transit.

Certificate Chain

A certificate chain is an ordered list of certificates. The certificates are ordered such that the server or user certificate is first, followed by the certificate of its CA.

CAs can either sign their own certificates (that is, they are self-signed) or they can be signed by another certificate authority. If they are self-signed, they are typically called root CAs. If they are not self-signed, they are typically called subordinate CAs or intermediate CAs.

If a user or server certificate was signed by a CA with a self-signed certificate, the certificate chain is composed of exactly two certificates: the end entity certificate and the root CA.

If a user or server certificate was signed by an intermediate CA, then the certificate chain is longer. The first two elements are still the end entity certificate, followed by the certificate of the intermediate CA. But the intermediate CA's certificate are then followed by the certificate of its CA. This listing then continues until the last certificate in the list is for a root CA. Thus, a certificate chain can be infinitely long. In practice, however, most certificate chains have only two or three certificates.

Trusted Roots

In order to validate a digital signature, you must trust at least one of the certificates in the user or server's certificate chain. You can directly trust the certificate of the user or server, or you can choose to trust any other certificate in the chain. Typically, the certificate that is trusted is the root CA's certificate.

Most application software that can use certificates already has a list of trusted certificates installed. These certificates are for root CAs and, hence, are called "trusted roots."Typically these CAs are commercial CAs. If you choose, you can add additional CAs to this list or remove CAs from the list.

Entry Rights Needed to Perform Tasks

The following list provides the specific entry rights an administrator needs to manage OpenText Certificate Server tasks within an OpenText eDirectory tree. These rights are the minimum entry rights needed.

This list should also be helpful to the administrator who wants to grant rights to another user to manage part or all of company's certificate authority and certificate management needs.

Table 25-2 *Administrator Entry Rights*

Tasks	Entry Rights Needed
Install OpenText Certificate Server	For the first installation to an eDirectory tree: ◆ Supervisor at the [Root] of the tree For subsequent installations: ◆ Supervisor to the W0 object ◆ Rights needed to create a Server Certificate object If a user doesn't have the rights to create a Server Certificate object, the installation finishes, but the Server Certificate objects need to be created manually by someone with the appropriate rights and applications that use these certificates need to be manually configured.
Creating an Organizational CA	◆ Supervisor on the Security container
Viewing the Organizational CA's properties and certificates	◆ Browse on the Organizational CA's object

Tasks	Entry Rights Needed
Exporting the Organizational CA's certificate(s)	◆ Browse on the Organizational CA's object
Issuing a public key certificate	◆ Read to the NDSPKI:Private Key on the Organizational CA's object However, if the object trying to issue the public key certificate is an NCP server, then the rights needed are: ◆ Write to the NDSPKI:Private Key on the Organizational CA’s object
Backing up and restoring an Organizational CA	◆ Supervisor on the Organizational CA's object
Moving the Organizational CA to a different server	◆ Supervisor on the Organizational CA's object
Validating the Organizational CA's Certificates	◆ Browse on the Organizational CA's object
Replacing the Organizational CA	◆ Supervisor on the Organizational CA's object
Deleting the Organizational CA	◆ Delete on the Organizational CA's object
Creating Server Certificate objects	◆ Supervisor on the server's container ◆ Read to the attribute NDSPKI:Private Key on the Organizational CA's object (only if using the Organizational CA) However, if the object trying to issue the public key certificate is an NCP server, then the rights needed are: ◆ Supervisor on the server’s container ◆ Write to the NDSPKI:Private Key on the Organizational CA’s object
Importing a public key certificate into a Server Certificate object	◆ Write to the attribute NDSPKI:Public Key Certificate on the Server Certificate object ◆ Write to the attribute NDSPKI:Certificate Chain on the Server Certificate Object
Deleting a Server Certificate object	◆ Delete on the Server Certificate object
Exporting a Trusted Root or Public Key Certificate from a Server Certificate object	◆ Browse on the Server Certificate object
Viewing the Server Certificate object's properties and certificates	◆ Browse on the Server Certificate object
Backing up and restoring a Server Certificate object	◆ Supervisor on the server object that owns the Server Certificate object to back-up ◆ Create on the Server object's container to restore.

Tasks	Entry Rights Needed
Validating Server Certificates	◆ Browse on the Server Certificate object
Revoking Server Certificates	◆ Read to the CA Private Key or Delete on the Server Certificate object or Supervisor on the Host Server (that is, the NCP™ Server object)
Replacing a server certificate's keying material	◆ Write to the attribute NDSPKI:PrivateKey on the Server Certificate object
Creating user certificates	◆ Read to the attribute NDSPKI:Private Key on the Organizational CA object ◆ Read and Write to the attribute NDSPKI:userCertificateInfo on the User object ◆ Read and Write to the attribute SAS:SecretStore on the User object ◆ Read and Write to the attribute userCertificate on the User object However, if the object trying to issue the public key certificate is an NCP server, then the rights needed are: ◆ Write to the NDSPKI:Private Key on the Organizational CA's object ◆ Read and Write to the attribute NDSPKI:userCertificateInfo on the User object ◆ Read and Write to the attribute SAS:SecretStore on the User object ◆ Read and Write to the attribute userCertificate on the User object
Importing a public key certificate into a User object	◆ Read and Write on the attribute NDSPKI:userCertificateInfo on the User object ◆ Read and Write to the attribute NDSPKI:userCertificate on the User object
Viewing a user certificate's properties	◆ Browse on the User object
Exporting a user certificate	◆ Browse on the User object
Exporting a user's private key and certificate	◆ You must be logged in as the user
Deleting a user certificate and private key	◆ Read and Write to NDSPKI:userCertificateInfo ◆ Read and Write to userCertificate
Validating User Certificates	◆ Browse on the User object

Tasks	Entry Rights Needed
Revoking User Certificates	◆ Read to the CA Private Key or Delete on the User Object or be logged-in as the User and Write to the userCertificate attribute
Creating a Trusted Root Container	◆ Create on the Security container
Creating a Trusted Root object	◆ Create on the Trusted Root Container in which the Trusted Root object will reside
Viewing a Trusted Root object's properties	◆ Browse on the Trusted Root object
Replacing a trusted root certificate	◆ Read and Write to NDSPKI:Not After on the Trusted Root object ◆ Read and Write to NDSPKI:Not Before on the Trusted Root object ◆ Read and Write to NDSPKI:Subject Name on the Trusted Root object ◆ Read and Write to NDSPKI:Trusted Root Certificate on the Trusted Root object
Validating a trusted root certificate	◆ Browse on the Trusted Root object
Revoking a trusted root certificate	◆ Read to the CA Private Key or Delete on the Trusted Root object
Deleting a Trusted Root object	◆ Delete on the Trusted Root object
Creating a CRL Container	◆ Supervisor on the Security container ◆ Write to the attribute ndspkiCRLContainerDN on the Organizational CA's object
Deleting a CRL Container	◆ Delete on the CRL container
Creating a CRL Configuration object	◆ Supervisor on the CRL container
Activating a CRL Configuration object	◆ Write to the attribute ndspkiCRLConfigurationDNList on the Organizational CA's object
Viewing and/or Modifying a CRL Configuration object's Properties	Modifying: ◆ Supervisor on the CRL Configuration object or ◆ Write to the attribute being modified on the CRL Configuration object Viewing: ◆ Browse on the CRL Configuration object
Deleting a CRL Configuration object	◆ Delete on the CRL Configuration object
Creating a CRL object	◆ Supervisor of the CRL Configuration object

Tasks	Entry Rights Needed
Exporting a CRL file	◆ Read from the attribute certificateRevocationList
Replacing a CRL file	◆ Browse on the CRL object
Viewing a CRL object's properties	◆ Browse to the attribute certificateRevocationList
Deleting a CRL object	◆ Delete on the CRL Distribution Point
Creating a Security container	◆ Create at the root of the OpenText eDirectory tree
Creating a SAS service object	◆ Supervisor on the object's container ◆ Write to the attribute SAS:Service DN on the server that the object is being created for.

26 Managing Passwords

This section provides an overview of Universal Password, password policies, and password self-service.

- "Understanding Universal Password" on page 707
- "Understanding Non-Reversible Password Storage" on page 710
- "Password Policies" on page 711
- "Deploying Universal Password" on page 711
- "Managing Passwords by Using Password Policies" on page 716
- "Security Considerations" on page 744
- "Importing Hash Based Passwords Into eDirectory" on page 745

Understanding Universal Password

Universal Password is managed by the Secure Password Manager, a component of the NetIQ Modular Authentication Services (NMAS) module. The Secure Password Manager simplifies the management of password-based authentication schemes across a wide variety of OpenText products as well as OpenText partner products. The management tools expose only one password and do not expose all of the behind-the-scenes processing for backwards compatibility.

Secure Password Manager and the other components that manage or make use of Universal Password are installed as part of an OpenText eDirectory. However, Universal Password is not enabled by default. Because all APIs for authentication and setting passwords are moving to support Universal Password, all the existing management tools, when run on clients with these new libraries, automatically work with the Universal Password.

NOTE: OpenText Identity Console supports the Universal Password. It also continues to support the NDS password for older systems in the network. After Universal Password has been configured and enabled for a user, the OpenText Identity Console has the capability of automatically upgrading/migrating the NDS password to the Universal Password.

How Secure Is Universal Password?

Reversible encryption of Universal Password is required for convenient interoperation with other password systems. Administrators have to evaluate the costs and benefits of the system. Using a Universal Password stored in OpenText eDirectory might be more secure or convenient than attempting to manage several different passwords. OpenText provides several levels of security to make sure Universal Password is protected while stored in OpenText eDirectory.

A Universal Password is protected by three levels of security:

- encryption of the password itself

- OpenText eDirectory rights
- file system rights

The Universal Password is encrypted by a user specific key. Both the Universal Password and the user key are stored in system attributes that only OpenText eDirectory can read. The user key is stored encrypted with the tree key, and the tree key is protected by a unique NICI key on each machine. Note that neither the tree key nor the NICI key is stored within OpenText eDirectory. They are not stored with the data they protect.

The tree key is present on each machine within a tree, but each tree has a different tree key. So, data encrypted with the tree key can be recovered only on a machine within the same tree. Thus, while stored, the Universal Password is protected by three layers of encryption.

Each key is also secured via OpenText eDirectory rights. Only administrators with the Supervisor right or the users themselves have the rights to change Universal Passwords.

File system rights ensure that only a user with the proper rights can access these keys.

By default, the user specific key and the tree key are 3 DES keys. OpenText eDirectory 9.2 supports AES 256-bit keys. To create an AES 256-bit key, see Creating an AES 256-Bit Tree Key in the NICI Administration Guide. As an administrator, you can re-encrypt passwords using the `diagpwd` utility. For more information, see "Universal Password Diagnostic Utility" on page 741.

NOTE: The Password policy should allow the user running this utility to retrieve the user's universal password.

If Universal Password is deployed in an environment requiring high security, you can take the following precautions:

1. Make sure that the following directories and files are secure:

Platform	Directories/Files
Windows	◆ `%SystemRoot%\SysWOW64\Novell\nici` ◆ `%SystemRoot%\System32\` where the NICI DLL is installed
Linux	◆ `/var/opt/novell/nici` ◆ `/etc/opt/novell/nici64.cfg` ◆ `/opt/novell/lib64/libccs2.so` and the NICI shared libraries in the same directory

   Consult the documentation for your system for specific details of the location of NICI and OpenText eDirectory files.

2. As with any security system, restricting physical access to the server where the keys reside is very important.

Universal Password

In the past, administrators have needed to manage multiple passwords (simple password, NDS password, enhanced password) because of password limitations. Administrators have also needed to deal with keeping the passwords synchronized.

- NDS Password: The older NDS password is stored in a hash form that is nonreversible. Only the NDS system can make use of this password, and it cannot be converted into any other form for use by any other system.
- Simple Password: The simple password was originally implemented to allow administrators to import users and passwords (clear text and hashed) from foreign nds-cluster-config directories such as Active Directory and iPlanet.

  The limitations of the simple password are that no password policy (minimum length, expiration, etc.) is enforced.
- Enhanced Password: The enhanced password is no longer supported by OpenText. The enhanced password is the forerunner of Universal Password. It offers some password policy, but its design is not consistent with other passwords. It provides a one-way synchronization and it replaces the simple or NDS password.

OpenText introduced Universal Password as a way to simplify the integration and management of different password and authentication systems into a coherent network.

Universal Password addresses these password problems by doing the following:

- Providing one password for all access to OpenText eDirectory.
- Enabling the use of extended characters in passwords.
- Enabling advanced password policy enforcement.
- Allowing synchronization of passwords from OpenText eDirectory to other systems.

Most features of password management require Universal Password to be enabled.

For detailed information, see “Deploying Universal Password” on page 711.

Password Policies

Universal Password provides the ability to create advanced password policies. A password policy is a collection of administrator-defined rules that specify the criteria for creating and replacing end user passwords. NMAS allows you to enforce password policies that you assign to users in OpenText eDirectory.

You manage password policies by using OpenText Identity Console.

For more information, see “Managing Passwords by Using Password Policies” on page 716.

Password Synchronization

Password synchronization across connected systems is a feature included with OpenText Identity Manager. It provides the following benefits:

- Bidirectional password synchronization
- Enforcement of Password Policies on connected systems

- E-mail notification when synchronization fails
- The ability to check password synchronization status for a user

For more information, see Connected System Support for Password Synchronization.

Understanding Non-Reversible Password Storage

Universal Passwords are stored in OpenText eDirectory after encryption and these passwords can be retrieved by OpenText eDirectory whenever required. For example, at the time of authentication.

As an alternative to Universal Password, OpenText eDirectory 9.2 supports storage of hashed passwords using Password Based Key Derivation Function 2 (PBKDF2) hashing algorithm (RFC 2898). If PBKDF2 hash of password is enabled, the user's passwords cannot be retrieved. For more information, see "Universal Password Configuration Options" on page 734.

IMPORTANT: OpenText eDirectory 9.2 onwards, in a password policy, if Universal Password is disabled, PBKDF2 hash of password gets enabled automatically. The existing password policies with Universal Password disabled, will not be enforced on the users before upgrading to OpenText eDirectory 9.2. But if you upgrade your server to eDirectory 9.2, these password policies will get enforced automatically to all the users in the tree. If you want to avoid this, remove all assignments of such password policies before upgrading.

If you are switching to PBKDF2 passwords from NDS passwords, you must also switch to the SCRAM login method manually. For more information on SCRAM login method, see "Password Authentication" on page 604.

NOTE: ◆Passwords created using PBKDF2 hashing algorithm, are case-sensitive unlike NDS passwords which are case in-sensitive.

- Passwords created with PBKDF2 hashing algorithm do not support `nspmXCharHistoryLimit` and `nspmXCharLimit` rules in the password policy.
- By default, PBKDF2 is configured to use SHA-256. However, this can also be configured to use SHA-384 and SHA-512 using the `nspmPBKDF2HashAlgorithm` attribute. While configuring this attribute, you can specify one of the following values in the same format:
  - sha256
  - sha384
  - sha512
- By default, PBKDF2 is configured to use iteration count of 1. The iteration count can be increased using the `nspmPBKDF2IterationCount` attribute. If you increase the iteration count, the ldap bind performance will degrade.
- SCRAM login method does not support appending OTP to password. If there are users in the tree who use NDS login method with hash-based OTP (HOTP), do not authorize the use of SCRAM login method for such users.

Enabling Non-Reversible Password Storage

1. Start OpenText Identity Console.
2. On the OpenText Identity Console home page > **Authentication Management** > **Password Policies**.
3. Click **Create Password Policy** +, to start the Password Policy Wizard.
4. Provide a name for the policy and click **Next**.
5. De-select the Universal Password to enable PBKDF2 password.
6. Complete the Password Policy Wizard.

Password Policies

Universal Password provides the ability to create advanced password policies. A password policy is a collection of administrator-defined rules that specify the criteria for creating and replacing end user passwords. NMAS allows you to enforce password policies that you assign to users in OpenText eDirectory.

You manage password policies by using OpenText Identity Console.

For more information, see "Managing Passwords by Using Password Policies" on page 716.

Deploying Universal Password

This section describes how to deploy and manage Universal Password.

Follow the instructions in sections 2.1 through 2.8 to deploy Universal Password:

- "Step 1: Identify Your Need for Universal Password" on page 712
- "Step 2: Make Sure Your Security Container Is Available" on page 712
- "Step 3: Verify That Your SDI Domain Key Servers Are Ready for Universal Password" on page 712
- "Step 4: Check the Tree for SDI Key Consistency" on page 714
- "Step 5: Enable Universal Password" on page 714
- "Backward Compatibility" on page 715
- "Password Administration" on page 715
- "Issues to Watch For" on page 715

Step 1: Identify Your Need for Universal Password

If you answer yes to any of the following questions, you should plan to deploy and use Universal Password:

- Do you plan to have international users access OpenText Web-based services or use the OpenText Client for Windows to access OpenText file and print services?
- Do you plan to use OpenText Identity Manager, with its enhanced password policy and password synchronization capabilities?

Step 2: Make Sure Your Security Container Is Available

NMAS relies on storing global policies to the OpenText eDirectory tree, which is effectively the security domain. The security policies must be available to all servers in the tree.

NMAS places the authentication policies and login method configuration data in the Security container that is created off the [Root] partition. This information must be readily accessible to all servers that are enabled for NMAS. The purpose of the Security container is to hold global policies that relate to security properties such as login, authentication, and key management.

OpenText eDirectory 9.0 and above provides security container caching. This feature caches the security container data on local servers so NMAS doesn't need to access the Security container with every attempted login. See the "Security Object Caching" on page 606.

With NMAS and eDirectory 8.8.*x* and later, we recommend that you create the Security container as a separate partition and that the container be widely replicated. This partition should be replicated as a Read/Write partition only on those servers in your tree that are highly trusted.

WARNING: Because the Security container contains global policies, be careful where writable replicas are placed, because these servers can modify the overall security policies specified in the OpenText eDirectory tree. In order for users to log in with NMAS, replicas of the User objects and security container must be on the NMAS server.

For additional information, see TID3393169.

Step 3: Verify That Your SDI Domain Key Servers Are Ready for Universal Password

You must verify that the SDI Domain Key servers meet minimum configuration requirements and have consistent keys for distribution and use by other servers within the tree. These steps are crucial. If you don't follow them as outlined, you could cause serious password issues on your system when you turn on Universal Password.

1 At a Windows server command prompt, run `sdidiag.exe`.

`sdidiag.exe` is not shipped with OpenText eDirectory. Once installed, run `sdidiag.exe`. The file is available as part of a security patch associated with TID 2974092.

2 Log in as an Administrator by entering the server (full context), the tree name, the user name, and the password.

3 Check to make sure all your servers are using 168-bit keys for 3DES tree key and 256-bit keys for AES 256-bit tree key.

   Follow the instructions in TID 3364214 to ensure that this requirement is met.

4 Enter the command CHECK `-v >> installation folder\sdinotes.txt`.

   The output to the screen displays the results of the CHECK command.

5 If no problems are found, go to "Step 4: Check the Tree for SDI Key Consistency" on page 714.

   or

   Follow the instructions written to the `installation folder\sdinotes.txt` file to resolve any configuration and key issues, then continue with Step 6.

6 Verify that the SDI Domain Key Servers are running NICI *3.0*.

   If the version is earlier, upgrade OpenText eDirectory to 9.0 or later, which upgrades NICI to 3.0 or later respectively:

7 (Optional) Re-run the SDIDIAG CHECK command. See Step 4.

For more information on using SDIDIAG, see TID 3364214.

Adding or Removing an SDI Domain Key Server

To remove a server as an SDI Domain Key Server, complete the following procedure:

1 `sdidiag.exe` is not shipped with OpenText eDirectory. `sdidiag.exe` can be downloaded from the Software License and Download (https://sld.microfocus.com) portal. Once downloaded, run sdidiag.exe.

2 Log in as an administrator with management rights over the Security container and the W0.KAP.Security objects by entering the server (full context), the tree name, the user name, and the password.

3 Enter the command RS `-s servername`.

   For example, if `server1` exists in container PRV in the organization `Novell` within the `Novell_Inc` tree, you would type `.server1.PRV.Novell.Novell_Inc.` for the servername.

To add a server as an SDI Domain Key Server, complete the following procedure:

1 From a Windows server, open a command prompt box and run `sdidiag.exe`.

2 Log in as an Administrator by entering the server (full context), the tree name, the user name, and the password.

3 Enter the command AS `-s servername`.

   For example, if `server1` exists in container PRV in the organization `Novell` within the `Novell_Inc` tree, you would type `.server1.PRV.Novell.Novell_Inc.` for the servername.

Step 4: Check the Tree for SDI Key Consistency

Verify that all instances of cryptographic keys are consistent throughout the tree. To ensure that each server has the cryptographic keys necessary to securely communicate with the other servers in the tree:

1. At a Windows server command prompt, run `sdidiag.exe`.
2. Enter the command `CHECK -v >> sys:system\sdinotes.txt -n container DN`.

   For example, if user Bob exists in container USR in the organization Acme within the Acme_Inc tree, you would type `.USR.Acme.Acme_Inc.` for the container distinguished name (DN).

   This reports if there are any key consistency problems among the various servers and the Key Domain servers.

   The output to the screen displays the results of the `CHECK` command.
3. If no problems are reported, you are ready to enable Universal Password. Go to "Step 5: Enable Universal Password" on page 714.

   or

   If problems are reported, follow the instructions in the `sdinotes.txt` file.

   In most cases, you are prompted to run the command `RESYNC -T`. This command can be repeated any time NMAS reports -1418 or -1460 errors during authentication with Universal Password.

   For more information on SDIDIAG options and operations, refer to the following:

   - TID 3364214
   - TID 7005397

Step 5: Enable Universal Password

1. Log into OpenText Identity Console.
2. On the OpenText Identity Console home page, click **Authentication Management** > **Password Policies**.
3. Click **Create Password Policy** +, to start the Password Policy Wizard.
4. Provide a name for the policy and click **Next**.
5. Click **Configuration** tab > click **General** > click **Enable Universal Password**.
6. Complete the Password Policy Wizard.

IMPORTANT: If you assign a policy to a container that is the root of a partition, the policy assignment is inherited by all users in that partition, including users in subcontainers. To determine whether a container is a partition root, browse for the container and note whether a partition icon is displayed beside it.

If you assign a policy to a container that is not the root of a partition, the policy assignment is inherited only by users in that specific container. It is not inherited by users that are in subcontainers. If you want the policy to apply to all users below a container that is not a partition root, you must assign the policy to each subcontainer individually.

Backward Compatibility

Universal Password is designed to supply backward compatibility to existing services. By default, passwords changed with this service can be synchronized to the simple and NDS passwords on the User object. You can choose which passwords you want to have synchronized by using the Password Policies under Authentication Management.

The exception to this is the use of international characters in passwords. Because the character translations are different for older clients, the actual values no longer match. We recommend that all OpenText Client software be upgraded in order for full, system-wide international passwords to function properly.

The NetWare Storage Management Services (SMS) infrastructure is used for OpenText and third-party backup and restore applications. The system passwords used by these OpenText and third-party products cannot contain extended characters if they are to function in a mixed environment.

NOTE: Refer to TID 3065822 to see which applications and services are Universal Password-capable, as well as which applications and services are extended character-capable. Many applications and services can use extended characters without Universal Password.

Password Administration

You can use the following methods to administer Universal Password:

- **OpenText Identity Console (Recommended):** Administering passwords by using OpenText Identity Console automatically sets the Universal Password to be synchronized to simple and NDS password values for backward compatibility. The OpenText Identity Console **Authentication Management** > **Password Policies** does allow for granular management of individual passwords and authentication methods that are installed and configured in the system.

  In OpenText Identity Console using the Password Management tile, you can use password policies to specify how Universal Password is synchronized with NDS, simple, and distribution passwords. In addition, an OpenText Identity Console task is provided that lets an Administrator set a user's Universal Password.

Issues to Watch For

- When you disable a user’s NDS password, the NDS password is set to an arbitrary value that is unknown to the user. The following list describes how some login methods handle this change:
  - The simple password method is not disabled if the NDS password is disabled. The simple password method uses the Universal Password if it is enabled and available. Otherwise, it uses the simple password. If Universal Password is enabled but not set, then the simple password method sets the Universal Password with the simple password.

- The enhanced password method is not disabled when the NDS password is disabled. The enhanced password does not use the Universal Password for login.
- The NDS password method (Universal Password) is not disabled when the NDS password is disabled. The NDS password method uses the Universal Password if it is enabled and available. Otherwise, it uses the NDS password. If the Universal Password is enabled but not set, then the NDS Password method sets the Universal Password with the NDS password.

- If an administrator changes a user's Universal Password, such as when creating a new user or in response to a help desk call, for security reasons the password is automatically expired if you have enabled the setting to expire passwords in the password policy. This is the **Number of days before password expires (0-365)** setting in the password policy under **Advanced Password Rules**. For this particular feature, the number of days is not important, but the setting must be enabled.

  NOTE: To overwrite this behavior, select the **Do not expire the user's password when the administrator sets the password** option in the password policy.

- If you create a password policy and enable Universal Password and enable Advanced Password Rules, the Advanced Password Rules are enforced instead of any existing password settings for NDS password. The legacy password settings are ignored. No merging or copying of previous settings is done automatically when you create password policies.

  For example, if you had a setting for the number of grace logins that you were using with the NDS password, when you enable Universal Password you need to re-create the grace logins setting in the Advanced Password Rules in the password policy.

  NMAS replaces the NDS password setting on the user object with corresponding password policy settings. For example, if the number of grace logins for the user object is 4, and it is 5 for the password policy, when the user logs in or changes the password, the number of grace logins for the user object changes to 5.

Managing Passwords by Using Password Policies

You can use password policies to increase security by setting rules for how users create their passwords. You can also decrease help desk costs by providing users with self-service options for forgotten passwords and for resetting passwords.

The following is discussed in this section:

- "Overview of Password Policy Features" on page 717
- "Planning for Password Policies" on page 717
- "Prerequisite Tasks for Using Password Policies" on page 721
- "Creating Password Policies" on page 722
- "Assigning Password Policies to Users" on page 739
- "Finding Out Which Policy a User Has" on page 740
- "Setting A User's Password" on page 740
- "Universal Password Diagnostic Utility" on page 741
- "Troubleshooting Password Policies" on page 742

Overview of Password Policy Features

A password policy is a collection of administrator-defined rules that specify the criteria for creating and replacing end-user passwords. NMAS enables you to enforce password policies that you assign to users in OpenText eDirectory.

Password policies can also include Forgotten Password Self-Service features, to reduce help desk calls for forgotten passwords. Another self-service feature is Reset Password Self-Service, which lets users change their passwords while viewing the rules the administrator has specified in the password policy. Users access these features through the Identity Manager User Application or OpenText Identity Console.

Using a password policy requires you to enable Universal Password for your users if you want to use advanced password rules, password synchronization, and many of the Forgotten Password features. For information on deploying Universal Password, see "Deploying Universal Password" on page 711.

You create password policies by using the Password Policy Wizard. In OpenText Identity Console, click **Authentication Management** > **Password Policies** > **Create Password Policy** ＋. For more information on creating password policies, see "Creating Password Policies" on page 722.

Planning for Password Policies

- "Planning How to Assign Password Policies in the Tree" on page 717
- "Planning the Rules for Your Password Policies" on page 717
- "Planning Login and Change Password Methods for your Users" on page 719

Planning How to Assign Password Policies in the Tree

We recommend that you assign a default policy to the whole tree and assign any other policies you use as high up in the tree as possible, to simplify administration.

NMAS determines which password policy is in effect for a user. See "Assigning Password Policies to Users" on page 739 for more information.

Planning the Rules for Your Password Policies

You can use the Advanced Password Rules in a password policy to enforce your business policies for passwords.

Keep in mind that the Novell Client (4.9.1), Identity Manager User Application, and the OpenText Identity Console display the password rules from the password policy. If your users will be changing their passwords through the LDAP server or on a connected system, you need to make the password rules readily available to users to help them be successful in creating a compliant password.

If you are using Identity Manager Password Synchronization, keep in mind that you must make sure that the users who are assigned password policies match with the users you want to participate in Password Synchronization for connected systems. Password policies are assigned with a tree-centric perspective. By contrast, Password Synchronization is set up per driver, on a per-server basis. To get the results you expect from Password Synchronization, make sure the users that are in a read/write or master replica on the server running the drivers for Password Synchronization match with the

containers where you have assigned password policies with Universal Password enabled. Assigning a password policy to a partition root container ensures that all users in that container and subcontainers are assigned the password policy.

Advanced Password Rules

Advanced Password Rules let you define the following criteria for the Universal Password:

- The lifetime of a password: Password policies provide the same policy features OpenText eDirectory has offered in the past, so you can specify how often a password must be changed and whether it can be reused.
- What a password contains: You can require a combination of letters, numbers, uppercase or lowercase letters, and special characters. You can exclude passwords that you don’t feel are secure, such as your company name. You can also require a certain number of characters in a password be “new,” unused in previous passwords, and configure the number of password policy violations allowed in a specified password.

To use Advanced Password Rules in a password policy, you must enable Universal Password. If you don't enable Universal Password for a policy, the password restrictions set for the NDS® password are enforced instead.

NOTE: When you create a password policy and enable Universal Password, the Advanced Password Rules are enforced, instead of any existing password settings for NDS Password. The legacy password settings are ignored. No merging or copying of previous settings is done automatically when you create password policies.

For example, if you have a setting for the number of grace logins that you use with the NDS Password, when you enable Universal Password you need to re-create the grace logins setting in the Advanced Password Rules in the password policy.

If you later disable Universal Password in the password policy, the existing password settings that you had are no longer ignored. They would be enforced for the NDS password.

NMAS 3.1 and later replaces the NDS password setting on the user object with corresponding password policy settings. For example, if the number of grace logins for the user object is 4, and it is 5 for the password policy, when the user logs in or changes the password, the number of grace logins for the user object changes to 5.

Enforcing Policies

When you assign a password policy to users in the tree, any password changes going forward must comply with the Advanced Password Rules in that policy. In Novell Client 4.9 SP2 or later, the rules are also displayed. In both methods of access, a noncompliant password is rejected. NMAS is the application that enforces these rules.

You can specify in the policy that existing passwords are checked for compliance and users are required to change existing noncompliant passwords. A password is marked as expired when the check for compliance option is enabled and the password does not satisfy the password policy rules.

You can also specify that when users authenticate through a portal, they are prompted to set up any Forgotten Password features you have enabled. This is called post-authentication services. For example, if you want users to create a Password Hint that can be e-mailed to them when they forget a password, you can use post-authentication services to prompt users to create a Password Hint at login time.

The post-authentication setting is the last option on the Forgotten Password property page.

Planning Login and Change Password Methods for your Users

There are several different ways a user can log in or change a password. For more information about upgrading to support Universal Password, see “Deploying Universal Password” on page 711.

This section explains the additional requirements for supporting Universal Password in each case:

- “Novell Client” on page 719
- “Identity Manager User Application and OpenText Identity Console” on page 720
- “Other Protocols” on page 720
- “Connected Systems” on page 721

Novell Client

If you are using the Novell Client, upgrade it to version 4.9 SP2 or later.

Keep in mind that using the Novell Client is not required, because users can log in through the OpenText Identity Console or other company portals depending on your environment. Also, the Novell Client is no longer required for Password Synchronization on Active Directory.

The following table describes the differences between Novell Client versions in regard to Universal Password and gives suggestions for handling legacy Novell Clients.

Table 26-1 *Universal Password with legacy Novell Clients*

Novell Client Version	Login	Change Password
Earlier than 4.9	Does not go through NMAS, so it does not support Universal Password. Instead, it logs in directly using the NDS password.	Changes the NDS Password directly, instead of going through NMAS. If you are using Universal Password, this can mean that the NDS password and the Universal Password are not kept synchronized. To prevent this, you have three options: ◆ Upgrade all the clients to version 4.9 or later. ◆ Block legacy clients from changing passwords by using an attribute value on a container. With this solution, legacy clients can still log in, but they cannot change the password. Password changes must be done using a later Novell Client or OpenText Identity Console. ◆ Use the password policy setting for **Remove the NDS Password when Setting Universal Password**. This is a drastic measure, because it prevents both login and password change through the NDS password.
4.9	Supports Universal Password.	Enforces password policy rules for Universal Password. If a user tries to create a password that is not compliant, the password change is rejected. However, the list of rules is not displayed to the user.
4.9 SP2 or later	Supports Universal Password.	Enforces password policy rules for Universal Password. In addition, it displays the rules to the users to help them create compliant passwords.

Identity Manager User Application and OpenText Identity Console

Identity Manager User Application and OpenText Identity Console provide Password Self-Service, so users can reset passwords and set up Forgotten Password Self-Service if the password policy provides it.

- We recommend that in your password policies you accept the default setting of **Synchronize NDS password when setting Universal Password**.

Other Protocols

Make sure that OpenText eDirectory, LDAP server, NMAS, and OpenText Identity Console are upgraded to support Universal Password.

For information about using AFP, CIFS, and other protocols with Universal Password, see “Deploying Universal Password” on page 711.

Connected Systems

If you are using Identity Manager Password Synchronization, make sure the following requirements are met so that user password changes are successful:

- Any Identity Manager drivers for the system have been upgraded to Identity Manager format.
- The Identity Manager driver configuration includes the new Password Synchronization policies.
- The Password Synchronization settings should specify that Universal Password is to be used, as well as the Distribution Password if bidirectional Password Synchronization is desired.
- Password filters have been deployed on the connected system to capture passwords, if necessary.

For more information, see "Connected System Support for Password Synchronization" (https://www.netiq.com/documentation/identity-manager-4.10/) in the *OpenText Identity Manager 4.10 Password Management Guide* (https://www.netiq.com/documentation/idenity-manager-4.10/).

Prerequisite Tasks for Using Password Policies

If you want to take advantage of all the features of password policies, you need to complete some steps to prepare your environment.

OpenText Identity Console application is available for download at the Software License and Download portal.

1 Upgrade your environment to support Universal Password.

  For more information, see "Deploying Universal Password" on page 711.

2 Upgrade your client environment to support Universal Password.

  See "Planning Login and Change Password Methods for your Users" on page 719 and "Deploying Universal Password" on page 711.

IMPORTANT: After you run the OpenText Identity Console application, the OpenText Identity Console runs in RAC mode. This means that administrators do not see any tasks unless they have assigned themselves to specific roles. Make sure you assign administrators to roles to give them access to all the OpenText Identity Console tasks.

IMPORTANT: If you upgrade to the latest version of the OpenText Identity Console without first upgrading OpenText eDirectory and then try to modify or create a password policy, OpenText Identity Console displays an error.

---

3 Configure the LDAP Group-Server object in OpenText eDirectory to require TLS for simple bind.

  This is the default setting when you configure OpenText Identity Console. Requiring TLS for simple bind is strongly recommended for Password Self-Service functionality, and is required for using the OpenText Identity Console > **Authentication Management** tile > **Password Policies**.

  If you are requiring TLS for simple bind, no additional configuration is needed for the LDAP SSL port.

IMPORTANT: If you choose not to require TLS for simple bind, this means that users are allowed to log in to the OpenText Identity Console by using a clear-text password.

You can use this option, but another step is required.

By default, the Password Self-Service functionality assumes that the LDAP SSL port is the one specified in the `System.DirectoryAddress` setting in the `PortalServlet.properties` file. If your LDAP SSL port is different, you must indicate the correct port by adding the following key pair to the `PortalServlet.properties` file:

`LDAPSSLPort=`*`your_port_number`*

For example, if you are running Tomcat, you would add this key pair in the `PortalServlet.properties` file in the `tomcat\webapps\nps\WEB_INF` directory.

4 To enable e-mail notification for Forgotten Password features, set up the SMTP server and customize the e-mail templates.

You are now ready to use all the features of password policies. Create policies as described in "Creating Password Policies" on page 722.

Creating Password Policies

Use the **Authentication Management** tile > **Password Policies** in OpenText Identity Console to create new password policies.

See the online help for information about each step in the wizard.

1 Log into OpenText Identity Console home page > **Authentication Management** > **Password Policies**.

2 Click **Create Password policy** +, to create a new password policy.

3 Follow the steps in the wizard to create Advanced Password Rules, Universal Password Configuration Options, and Forgotten Password selections for the policy.

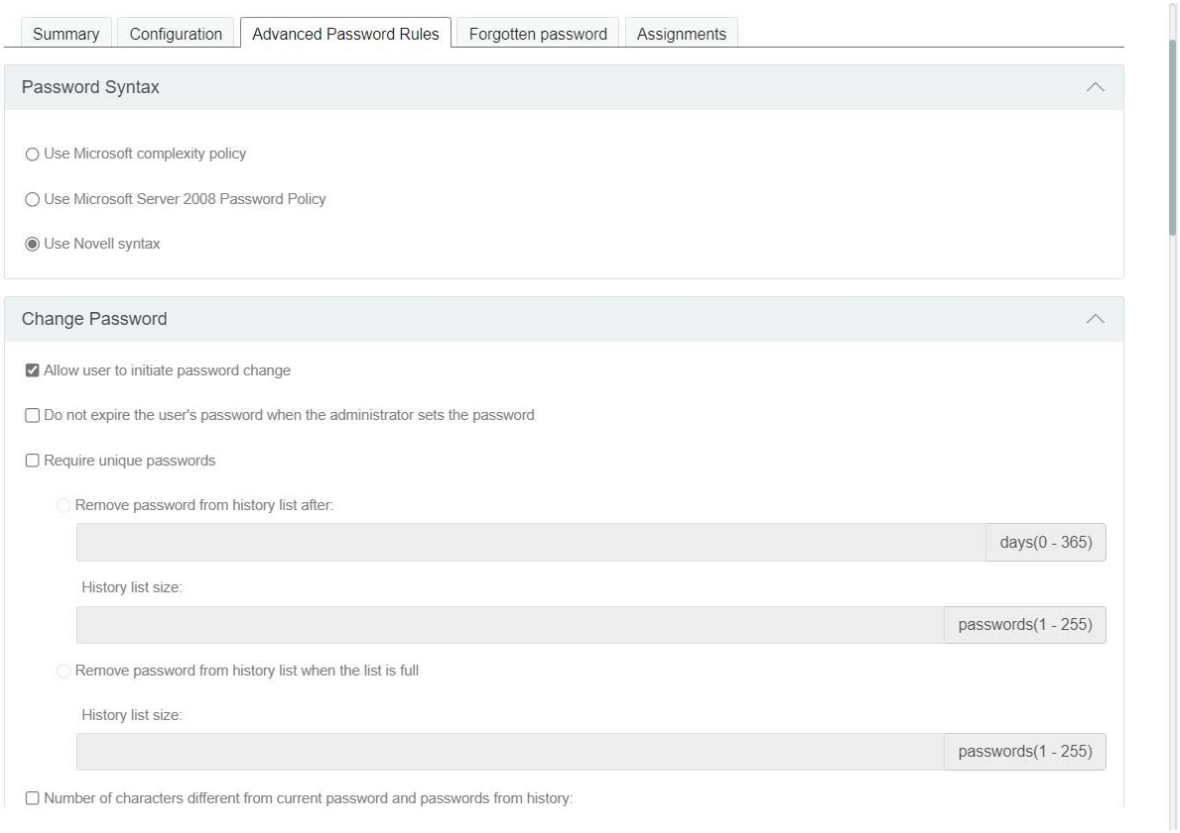
4 Assign the password policy to individuals, organizations, or your entire company, as necessary.

5 Review the settings for the new policy and click **Create**, then click **OK**.

Advanced Password Rules

Figure 26-1 shows the first section of the advanced password rules:

Figure 26-1 *Advanced Password Rules*



Password Syntax

You can specify one of three password syntax options to use for a password policy:

- **Use Microsoft complexity policy**
- **Use Microsoft Server 2008 Password Policy**
- **Use Novell syntax**

WARNING: OpenText Identity Console allows you to create a policy using the Microsoft Server 2008 Password Policy type, regardless of the version of NMAS installed on your server. However, you must have NMAS 3.3.4 or later installed to use this option. If you have a previous version of NMAS installed, the new password policy does not function properly.

- **Use Microsoft complexity policy**

  This setting allows you to use the Microsoft* Complexity Policy requirements. Use this option if you must synchronize passwords between OpenText eDirectory and Microsoft Active Directory.

  If you select this option for a policy, all users to which the policy is assigned must create passwords that meet the criteria of the Microsoft Complexity Policy as implemented in Universal Password. The criteria include:

  - Minimum password length is 6 characters.
  - Maximum password length is 128 characters.

- The password must contain at least one character from three of the four types of character, uppercase, lowercase, numeric, and special:
  - Uppercase characters - all uppercase characters in the Basic Latin and the Latin-1 character sets.
  - Lowercase characters - all lowercase characters in the Basic Latin and the Latin-1 character sets.
  - Numeric characters - 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
  - Special characters - all other characters.
- The values of the following user attributes can not be contained in the password: `CN`, `Given Name`, `Surname`, `Full Name`, and `displayName`.
- The password cannot contain the full value of the `CN` user attribute for the OpenText eDirectory account. NMAS does not perform this check if the length of the attribute is less than three characters.

- **Use Microsoft Server 2008 Password Policy**

  This setting allows you to use the Microsoft* Windows Server 2008 password policy complexity requirements. Use this option if you must synchronize passwords between OpenText eDirectory and Microsoft Active Directory.

  If you select this option for a policy, all users to which the policy is assigned must create passwords that meet the criteria of the Microsoft Windows Server 2008 Complexity Policy as implemented in Universal Password. If you select this option, several options on the Advanced Password Rules page are set to meet the criteria of the Complexity Policy. The criteria include:

  - Minimum password length is 7 characters, by default. You can configure the minimum password length in your environment using the **Minimum number of characters in password (1-512)** option. For more information about configuring the minimum number of characters, see "Password Length" on page 729.
  - Maximum password length is 512 characters.
  - The password must contain at least one character from three of the five types of character, uppercase, lowercase, numeric, non-alphanumeric characters, and other characters:
    - Uppercase characters - all uppercase European-language characters, with diacritical marks, as well as Greek and Cyrillic characters.
    - Lowercase characters - all lowercase European-language characters, with diacritical marks, as well as Greek and Cyrillic characters.
    - Numeric characters - 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
    - Non-alphanumeric characters - any of the following special characters: ( ) ` ~ ! @ # $ % ^ & * - + = | \ { } [ ] : ; " ' < > , . ? / _.
    - Other characters - any Unicode character that is categorized as an alphabetic character but is not uppercase or lowercase. This includes Unicode characters from Asian languages.
  - The password cannot contain any word from the list of excluded passwords. NMAS does not perform this check if the length of the excluded password is less than three characters. For more information about excluding passwords, see "Password Exclusions" on page 728.

- The password cannot contain the full value of the CN attribute or full or any part of the value of the `Full Name` attribute for the account, if the attribute contains at least three characters and is a single word. A part of the attribute value is defined as three or more consecutive characters delimited on both ends by the following characters: commas; periods; dashes; hyphens; underscores; spaces; pound signs; or tabs.

  NOTE: While using the Microsoft 2008 Password Policy, the CN and the `displayName` attributes are considered to be similar to the `samAccountName` and the `displayName` rule in AD.

- The maximum number of complexity policy violations allowed in a password is 2 by default. You can configure the number of complexity violations allowed using the **Maximum number of complexity policy violations in password (0-5)** option. For more information about configuring the maximum violations allowed, see "Password Complexity Violations" on page 730.

- **Use Novell syntax**

  This allows you to use the Novell syntax for the password policy. This option is selected by default. Standard settings for policies using Novell syntax include:

  - Minimum password length is 4 characters, by default. You can configure the minimum password length in your environment using the **Minimum number of characters in password (1-512)** option. For more information about configuring the minimum number of characters, see "Password Length" on page 729.
  - Maximum password length is 12 characters, by default. You can configure the maximum password length in your environment using the **Maximum number of characters in password (1-512)** option. For more information about configuring the maximum number of characters, see "Password Length" on page 729.

Password Syntax Precedence

If you modify the attributes of a password policy using Directory Administration or LDAP, outside of the OpenText Identity Console Password Policies, you may set up a conflict between one or more of the password policy types. For example, you could use LDAP to enable both the Microsoft complexity policy and Microsoft Windows 2008 Password Policy types for the same policy.

In the event of a conflict, OpenText eDirectory uses the following order of precedence:

- Microsoft Windows 2008 Password Policy
- Microsoft complexity policy
- Novell syntax

For more information about modifying password policies outside of the Password Management interface, see "Modifying Password Policies Outside of the Password Policies Interface" on page 733.

Change Password

- **Do not expire the user's password when the administrator sets the password**

  This option requires the user to go and change his or her password. This feature allows you to override the default. The default behavior in OpenText eDirectory, when password expiration is set, is to expire the user's password when the administrator sets the password.

- **Require unique passwords**

  When this option is selected, the user is prevented from changing the password to one that is already in the history list. If a user tries to change the password and reuse one that is in the history list, the password policy rejects the password and the user is prompted to specify a different one.

  You can specify how unique passwords are enforced by using one of the following two values:

  - **Remove password from history list after a specified number of days (0-365)** and a specified **History list size (1-255)**.

    If you require unique passwords, you can specify how many days a previous password remains stored in the history list for comparison.

    For example, if you specify a limit of 30 days, and the user's previous password was “mountains99,” that password remains in the history list for 30 days. During that time, if the user tries to change his or her password and reuse “mountains99,” the password policy rejects that password, and the user is prompted to specify a different one. After the 30-day period, the old password is no longer stored for comparison, and the password policy allows it to be reused.

    If you require unique passwords, you can also indicate how many passwords are stored in the history list for comparison. For example, if you specify 3, then the user's previous three passwords are stored. If a user tries to change his or her password and reuse one that is in the history list before the number of days specified for removal from the history list, the password policy rejects the password, and the user is prompted to specify a different one.

    NOTE: ◆If the **Use Microsoft Server 2008 Password Policy** option is selected, the **Require unique passwords** option is also selected by default.

    - If **Require unique passwords** is selected and you select **Remove password from history list after a specified number of days (0-365)** but don’t specify a number of days, the password is on the history list for 8 times the value set in the **Number of days before password expires (0-365)** field, in the Password Lifetime section. If neither field has a value, the password is on the history list for 365 days.
    - If you specify a password history list size and a number of days, and the number of passwords in the password history list size has been met, the user cannot change his or her password unless the password has expired. An administrator can change or set a user password even if the password list size has been met.
    - After one or more passwords expire in the password history list, the list is no longer full, and a user is again able to change his or her password. This limitation is included to prevent users from changing their passwords so many times that a password is no longer included in the password history list, and they can re-use it.
    - If a password history list size is not specified, the password history is never full.
    - When comparing a specified password against previous passwords in the password history, OpenText eDirectory differs from Active Directory. If the size of the password history list is “N,” Active Directory compares a specified password against “N” previous passwords. However, OpenText eDirectory compares a specified password against “N+1” previous passwords.

  - **Remove password from history list when the list is full** and the number of passwords reaches the specified **History list size** (1-255).

If you require unique passwords, you can indicate how many passwords are stored in the history list for comparison. This option works on a first-in, first-out basis, where the oldest passwords are removed from the history list first. For example, when a user creates a new password that is not currently in the history list, the oldest password in the history list is removed if the history list is full.

NOTE: ◆If the **Use Microsoft Server 2008 Password Policy** option is selected, the **Remove password from history list when the list is full** option is also selected by default. With the Microsoft Server 2008 syntax enabled, the **History list size** range is 0-24 passwords.

- If this option is selected, you should also select both the **Number of days before password can be changed** and **Number of days before password expires** options, with at least the minimum number of days for each.
- If you specify a password history list size of 0, NMAS only compares any new password created by a user against that user's current password.

- **Number of characters different from current password and passwords from history (0-6)** and a specified number of characters.

  When this option is selected, the user must specify a password that includes at least as many "new" characters, characters unused in previous passwords, as specified in the setting. This option is selected by default.

  You can specify how unique the unused characters must be by using the following value:

  - **Number of passwords in history to be considered for character exclusion (0-10)** and a specified number of characters

    If you require a certain number of unused characters for any new password, you can specify how many previous passwords to consider when checking a password for previously-used characters.

    For example, if you specify a minimum of three new characters and specify that five previous passwords should be considered for character exclusion, and a user creates the new password "mountains99," that password must include at least three characters not in any of the previous five passwords. If the user's password two changes previous was "maintains99," only two characters different from the new password, the password policy rejects that password, and the user is prompted to specify a different one.

NOTE: ◆Both the **Number of characters different from current password and passwords from history (0-6)** and **Number of passwords in history to be considered for character exclusion (0-10)** options are selected by default. However, the values of both options are set to 0 by default.

- If the value of the **Number of characters different from current password and passwords from history (0-6)** option is set to 0, the option is disabled.
- If the value of the **Number of passwords in history to be considered for character exclusion (0-10)** option is set to 0, only the current password is considered when OpenText eDirectory checks for "new" characters.
- These options require Universal Password to be enabled in the password policy.

Password Lifetime

- **Number of days before password can be changed (0-365)**

This option restricts the user from changing their Universal Password before the specified time has elapsed. For example, if this value is set to 30, a user must keep the same password for 30 days before he or she can change it.

- **Number of days before password expires (0-365)**

  This option causes a user's password to expire after a specified time has elapsed. For example, if this value is set to 90, a user's password expires 90 days after it has been set. If you enable grace logins, the user can log in with the expired password the specified number of times. Also, if you have not selected the Limit Grace Logins option, unlimited grace logins are allowed.

NOTE: ◆If the **Use Microsoft Server 2008 Password Policy** option is selected, the **Number of days before password can be changed** and **Number of days before password expires** options are also selected by default. With the Microsoft Server 2008 syntax enabled, the range for both options is 0-999 days.

- If an administrator changes a user's password, such as when creating a new user or in response to a help desk call, the password is automatically expired if you have enabled the setting to expire passwords in the password policy. For this particular feature, the number of days is not important, but this setting must be enabled. Selecting the **Do not expire the user's password when the administrator sets the password** option overrides this security enhancement.

- **Limit the number of grace logins allowed (0-254)**

  When the password expires, this value indicates how many times a user is allowed to log in to OpenText eDirectory by using the expired password. If grace logins are not enabled, the user cannot log in after a password has expired, and he or she requires administrator assistance to reset the password. If the value is 1 or more, the user has a chance to log in additional times before being forced to change the password. However, if the user does not change the password before all the grace logins are used, he or she is locked out and is unable to log in to OpenText eDirectory. Also, if you have not selected the **Limit the number of grace logins allowed** option, unlimited grace logins are allowed.

Password Exclusions

- **Exclude the following passwords**

  This allows you to manually specify the passwords you want to exclude. You can use this option to exclude specific words or single characters, not a pattern or an OpenText eDirectory attribute. You can also exclude passwords containing a specific special character, including a *, +, %, or space character. For example, if you add the character * to the list of excluded passwords, a user who tried to specify the password "Pa55w0rd*!" would receive an error saying that the specified password is invalid. This can be useful if you need to restrict users from specifying passwords containing special characters that cause issues with applications in your environment.

  For NMAS 3.1.3 and later, the strings in the exclude list cannot be contained in the password, and the comparison is case-insensitive. For example, if "test" is in the exclude list, then the following cannot be passwords: Test, TEST, ltest, test1, and latest.

  Keep in mind that password exclusions can be useful for a few words that you think would be security risks. Although an exclusion list feature is provided, it is not intended to be used for a long list of words, such as a dictionary. Long lists of excluded words can affect server

performance. Instead of a long exclusion list to protect against “dictionary attacks” on passwords, we recommend that you use the Advanced Password Rules to require numbers to be included in the password.

- **Exclude passwords that match attribute values**

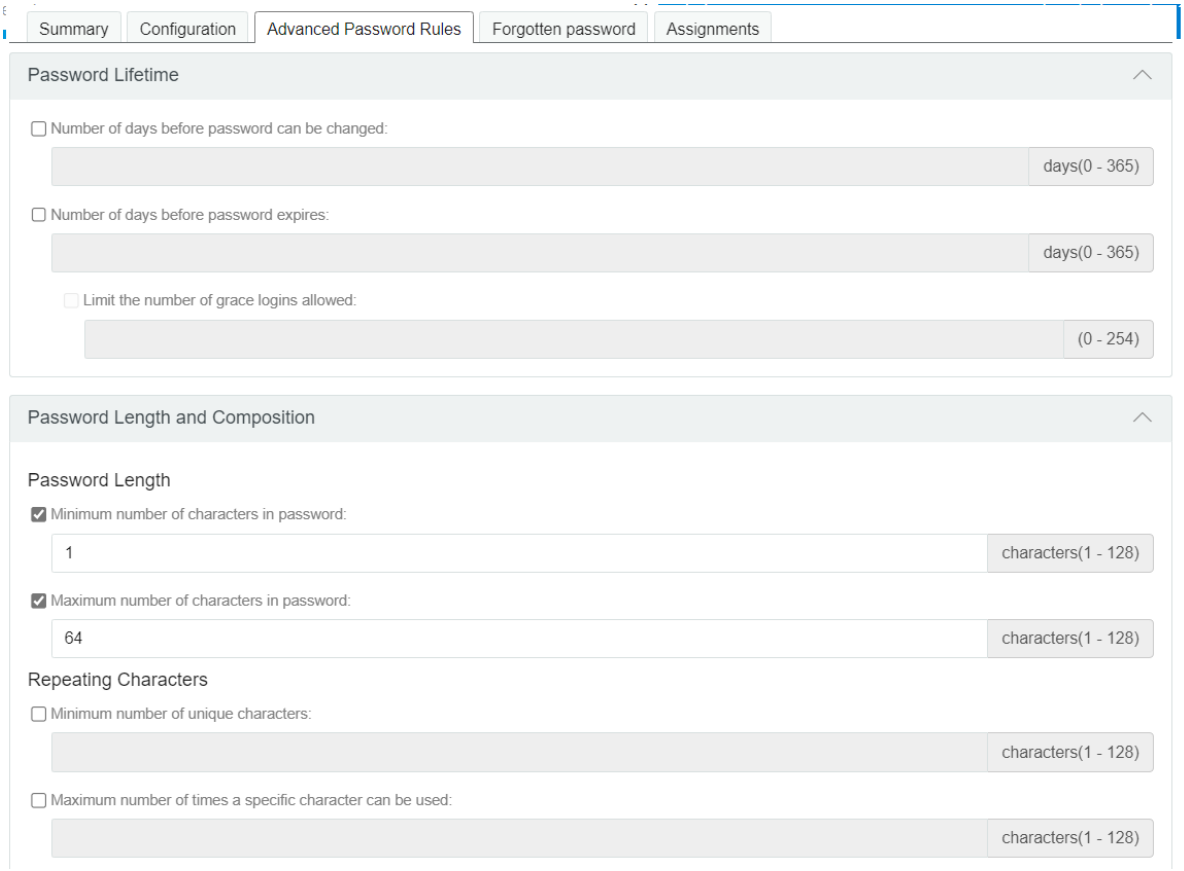
  This allows you to select User object attributes that you want to exclude from being used as passwords. For example, if you add the Given Name attribute to the list, and the Given Name attribute contained the value of Frank, then neither frank, frank1, nor 1frank could be used as the password.

  NMAS does not perform this check if the length of the excluded password is less than three characters.

  Use the plus and minus buttons to add and delete attribute values from the list.

NOTE: If the **Use Microsoft complexity policy** option is selected, the **Exclude passwords that match attribute values** option is also selected by default. With the Microsoft complexity policy syntax enabled, the list of attribute values to match is prepopulated with the following attributes: `Common name`; `Display name`; `Full name`; `First name`; and `Last name`.

Figure 26-2 *Advanced Password Rules Continued*



Password Length

- **Minimum number of characters in password (1-512)**
- **Maximum number of characters in password (1-512)**

NOTE: ◆The maximum length for any password created using NMAS is 512 characters.

- If the **Use Microsoft complexity policy** option is selected, neither the **Minimum number of characters in password** nor **Maximum number of characters in password** option is available.
- If the **Use Microsoft Server 2008 Password Policy** option is selected, only the **Minimum number of characters in password** option is available. The option is selected by default.
- If the **Use Novell syntax** option is selected, both the **Minimum number of characters in password** and **Maximum number of characters in password** options are also selected by default.

Password Complexity Violations

- **Maximum number of complexity policy violations in password (0-5)**

  This option allows you, as an administrator, to configure the number complexity policy violations you want to allow in passwords in your environment. By default, the Microsoft Server 2008 Password Policy requires that a password include at least one character from three of the five types of character, uppercase, lowercase, numeric, non-alphanumeric characters, and other characters. Therefore, the default number of violations allowed is 2. For more information on policy requirements for Microsoft Server 2008 Password Policy, see "Password Syntax" on page 723.

  However, if you want to make your password policy more or less restrictive, you can modify the default number of violations allowed. For example, if you change the default setting to 1, all passwords must include at least one character from four of the five character types listed above. If the setting is 4, passwords must include a character from only one of the five character types.

NOTE: The **Maximum number of complexity policy violations in password (0-5)** option is only available if you select the **Use Microsoft Server 2008 Password Policy** option. The option is selected by default.

Repeating Characters

- **Minimum number of unique characters (1-512)**
- **Maximum number of times a specific character can be used (1-512)**
- **Maximum number of times a specific character can be repeated sequentially (1-512)**

NOTE: If either the **Use Microsoft complexity policy** or **Use Microsoft Server 2008 Password Policy** options is selected, the **Minimum number of unique characters**, **Maximum number of times a specific character can be used**, and **Maximum number of times a specific character can be repeated sequentially (1-512)** options are unavailable.

Case Sensitive

In OpenText eDirectory, you can use the **Allow the password to be case sensitive** option to make your passwords case sensitive for all the clients that are upgraded to OpenText eDirectory 9.2.

NOTE: ◆The **Allow the password to be case sensitive** option is only available if you select the **Use Novell syntax** option. The option is selected by default.

- If you have opted to disable the Universal Password, Case Sensitive option will be checked and disabled by default.

The **Allow the password to be case sensitive** option is only available if you select the **Use Novell syntax** option. The option is selected by default.

With **Allow the password to be case sensitive** selected, you have four options:

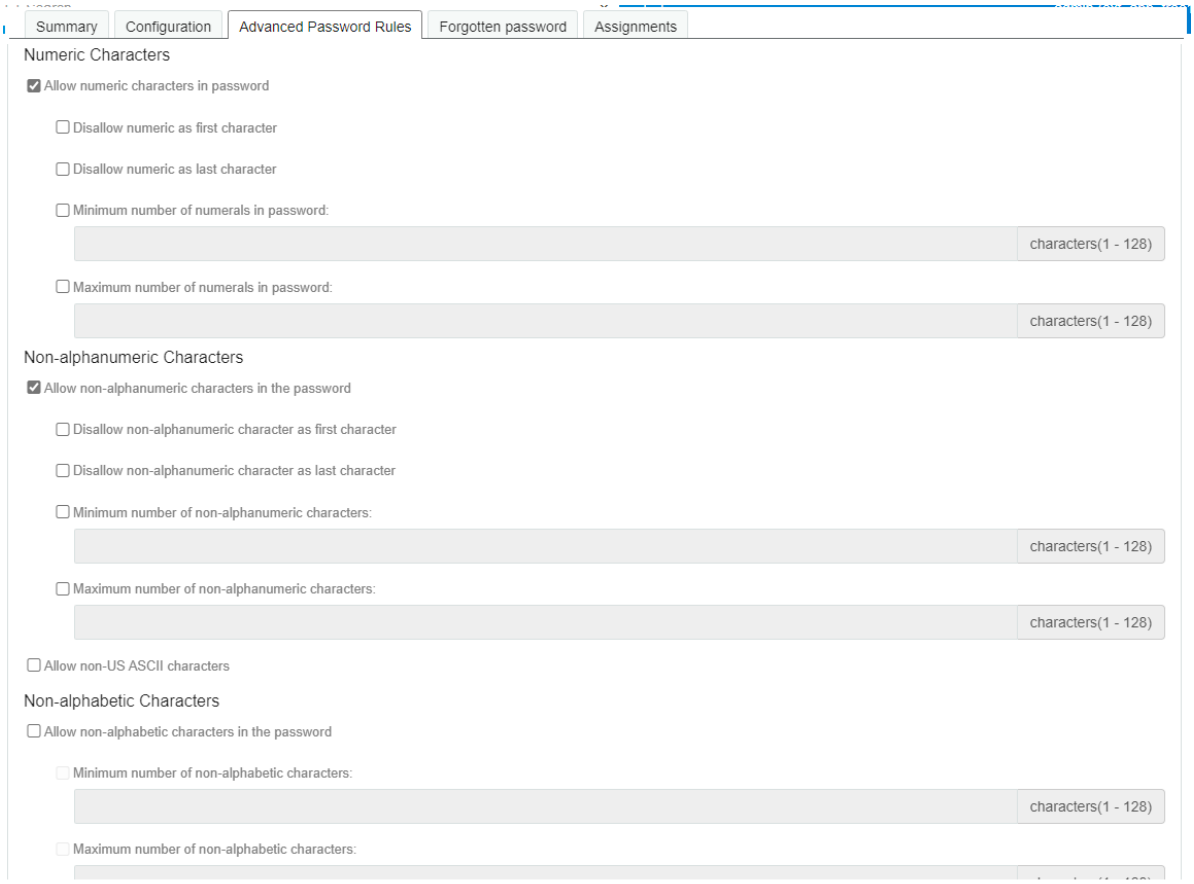
- **Allow the password to be case sensitive**
  - **Minimum number of upper case characters required in the password (1-512)**
  - **Maximum number of upper case characters allowed in the password (1-512)**
  - **Minimum number of lower case characters required in the password (1-512)**
  - **Maximum number of lower case characters allowed in the password (1-512)**

When **Allow the password to be case sensitive** is not selected, the passwords are case insensitive, and you have two options:

- **Minimum number of alphabetic characters allowed in password (1-512)**
- **Maximum number of alphabetic characters allowed in password (1-512)**

IMPORTANT: Passwords are stored with case, and are synchronized between systems with case sensitivity, even though the **Allow passwords to be case sensitive** option is not selected. The case of password characters is ignored if the **Allow the password to be case sensitive** option is not selected.

Figure 26-3 *Advanced Password Rules Final*



Numeric Characters

- **Allow numeric characters in password**
  - **Disallow numeric as first character**
  - **Disallow numeric as last character**
  - **Minimum number of numerals in password (1-512)**
  - **Maximum number of numerals in password (1-512)**

NOTE: The **Allow numeric characters in password** option is only available if you select the **Use Novell syntax** option. The option is selected by default.

Non-alphanumeric Characters

Non-alphanumeric characters are characters that are not numbers (0-9) or alphabetic characters. Alphabetic characters are defined as a-z, A-Z, and alphabetic characters in the Latin-1 code page 850.

- **Allow non-alphanumeric characters in the password**
  - **Disallow non-alphanumeric character as first character**
  - **Disallow non-alphanumeric character as last character**

- **Minimum number of non-alphanumeric characters (1-512)**
- **Maximum number of non-alphanumeric characters (1-512)**

- **Allow non-US ASCII characters**

  This option allows a password to include characters outside of the Basic Latin character set, also known as extended characters.

NOTE: The **Allow non-alphanumeric characters in the password** option is only available if you select the **Use Novell syntax** option. The option is selected by default.

Non-alphabetic characters

Non-alphabetic characters are the characters that are not alphabetic characters. Alphabetic characters are defined as a-z, A-Z, and alphabetic characters in the Latin-1 code page 850.

- **Allow non-alphabetic characters in the password**
  - **Minimum number of non-alphabetic characters (1-512)**
  - **Maximum number of non-alphabetic characters (1-512)**

NOTE:
- The **Allow non-alphabetic characters in the password** option is only available if you select the **Use Novell syntax** option.
- If you use the **Allow non-alphabetic characters in the password** option, ensure your policy does not unduly restrict possible passwords. For example, you can create a policy that requires multiple non-alphabetic characters or numerals but also *limits* the number of non-alphabetic characters allowed.

Modifying Password Policies Outside of the Password Policies Interface

In addition to creating, modifying, and assigning password policies using the OpenText Identity Console Authentication Management tile, you can modify policies outside of the Password Policies interface in one of the following ways:

- Modify the policy object directly using the Directory Administration interface.
- Modify the policy object directly using the ldapmodify command line tool.

However, it is not recommended that you manipulate password policies outside of the Password Policies interface, as this manipulation might cause issues in your environment if all attributes are not properly set. If you set multiple policy types for a single policy, for example, only the “highest” policy type in the order of precedence takes effect, and OpenText eDirectory ignores any policy rules for the “lower” policy types applied. For more information about password policy type precedence, see “Password Syntax Precedence” on page 725.

In addition, if you change the type of a password policy from the Microsoft Server 2008 Password Policy type to the Microsoft complexity policy type without using the Password Policies interface, OpenText Identity Console does not delete the existing Microsoft Server 2008 Password Policy attribute (`nspmAD2K8Syntax`) in the policy object. Instead, OpenText Identity Console sets the value of the attribute to `False`. In this situation, OpenText eDirectory ignores all policies and rules set for either policy type.

Another issue can occur when you use LDAP to modify specific rules for a policy. If you modify a policy so that two rules conflict, OpenText eDirectory applies a rule that is selected or is set to `True` in the policy instead of a conflicting rule that is not selected or is set to `False`.

For example, you can create a policy and then modify that policy to both not allow numeric characters and allow non-alphabetic characters. Because the value of the `nspmNonAlphaCharactersAllowed` attribute is set to `True`, all non-alphabetic characters are allowed, including numeric characters, even though the `nspmNumericCharactersAllowed` is set to `False`.

Random Password Generation

Instead of specifying a particular password, users can also request a randomly-generated password. Randomly-generated passwords automatically conform to the complexity requirements and other restrictions of the password policy assigned to the user.

Randomly-Generated Microsoft Server 2008 Passwords

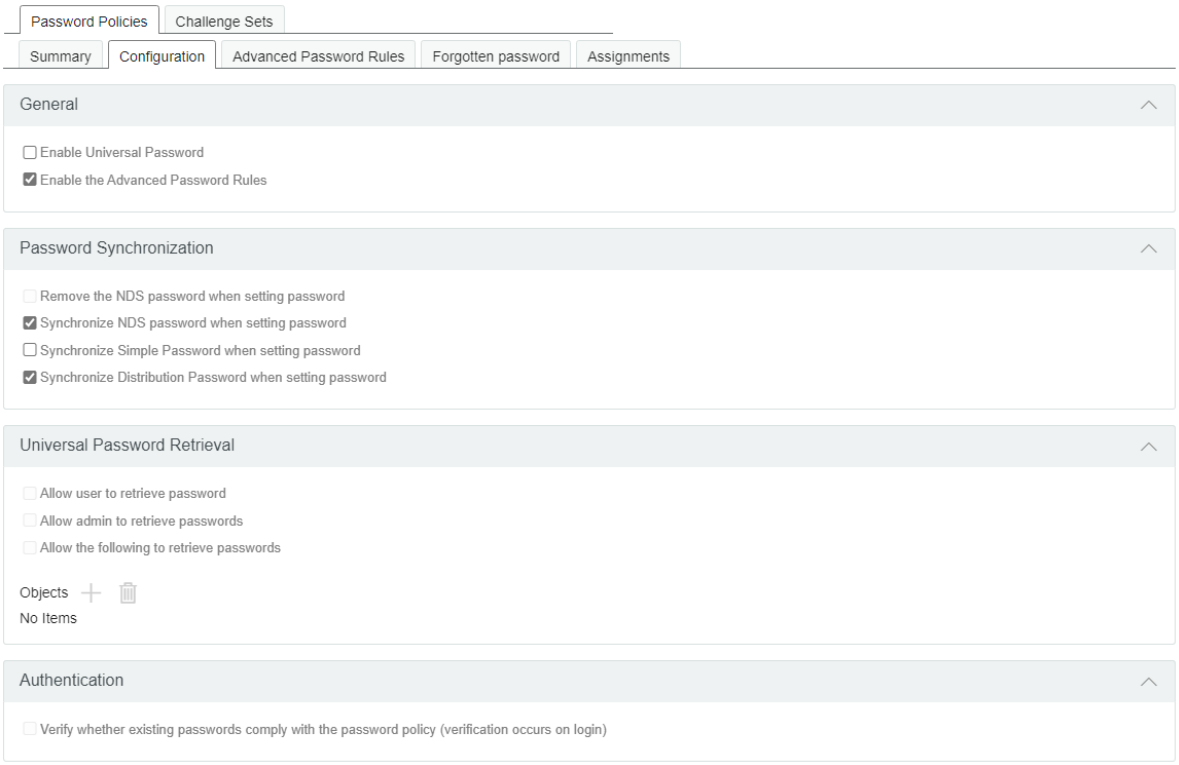
Randomly-generated passwords for Microsoft Server 2008 Password Policy policies differ in the following ways from randomly-generated passwords using other password policy types:

- If a user is assigned a password policy that uses the Microsoft Server 2008 Password Policy type and requests a randomly-generated password, NMAS generates the password based on the number of password complexity violations allowed for the policy.
- If the number of password complexity violations allowed is set to the maximum value of 5, any randomly-generated password consists only of uppercase or lowercase alphabetic characters.
- If the configured password complexity requirements are extremely strict, even randomly-generated passwords may not be valid for the password policy.
- The maximum length of any randomly-generated Microsoft Server 2008 Password Policy password is 16 characters, unless the minimum length configured in the policy is more then 16 characters. If the minimum length is more than 16, the length of the generated password is the minimum length set in the policy. For example, if the minimum length of a password is set to 20 characters using a Microsoft Server 2008 policy, the randomly-generated password is always 20 characters long.

Universal Password Configuration Options

The following figure shows an example of the Universal Password configuration options:

Figure 26-4 *Configuration Options*



- **Enable Universal Password**

  Enables Universal Password for this policy. You can choose to either enable or disable Universal Password.

- **Enable the Advanced Password Rules**

  Enables the Advanced Password Rules found on the Advanced Password Rules page for this policy. These advanced password rules help secure your environment by giving you control over password lifetime and what the password can contain.

- **Password Synchronization**

  - **Remove the NDS password when setting Universal Password**

    If this option is selected, the NDS password is disabled when the Universal Password is set. Also, when the NDS password is set, the NDS password hash is set to a random value that is not known except to OpenText eDirectory. There might or might not be a password that could be hashed to the random value.

  - **Synchronize NDS password when setting Universal Password**

    If this option is selected, and the Universal Password is set, the NDS password is set at the same time and with the same password.

  - **Synchronize Simple Password when setting Universal Password**

    NOTE: The setting of this option does not affect your ability to import user passwords by using ICE.

If this option is selected, and the Universal Password is set, the Simple Password is set at the same time and uses the same password.

- **Synchronize Distribution Password when setting Universal Password**

  Determines whether the Identity Manager Metadirectory engine can retrieve or set a user's Universal Password in OpenText eDirectory.

  If this option is selected, and the Universal Password is set, the Distribution Password is set at the same time and uses the same password.

  The Distribution Password can be used with Identity Manager to perform password synchronization to connected systems. This option also allows the Metadirectory engine to retrieve a user's Universal Password in OpenText eDirectory.

- **Universal Password Retrieval**

  NOTE: If you have opted to disable Universal Password, the following options will be disabled by default.

  - **Allow user to retrieve password**

    Determines whether the Forgotten Password Self-Service feature can retrieve a password on behalf of a user, so that the password can be e-mailed to the user. If this option is not selected, the corresponding feature is dimmed on the Forgotten Password page in the Password Policy.

    This option allows users to retrieve their own passwords by using NMAS LDAP extensions.

  - **Allow admin to retrieve passwords**

    Lets you retrieve users' passwords by using a third-party product or service that uses this functionality.

    This option is not recommended. Instead, you should use the **Allow the following to retrieve passwords** option to assign password read rights to specific objects, such as the SAMBA or freeRADIUS service objects, that need this ability to perform their functions.

    If **Allow admin to retrieve passwords** is selected, then users that have write privileges on the target object's ACL attribute can retrieve the target object's password.

  - **Allow the following to retrieve passwords**

    Lets you insert an object that has the ability to retrieve passwords.

    NOTE: Members with insufficient privileges receives a -672 error while using the **Check Password Status** task on any given user.

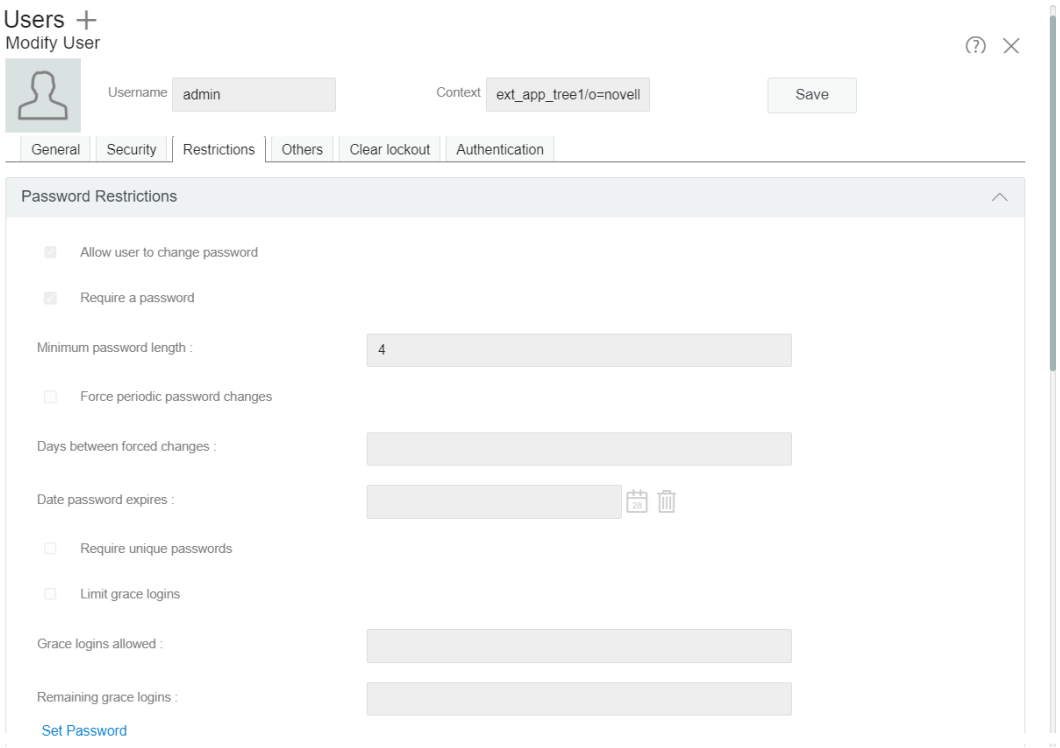
- **Authentication**

  - **Verify whether existing passwords comply with the password policy (verification occurs on login)**

    If this option is selected, and users log in through **OpenText Identity Console**, their existing passwords are checked to make sure they comply with the Advanced Password Rules in the users' password policy. If an existing password does not comply, users are required to change it. If Universal Password is disabled, this option will also be disabled by default.

    An administrator can change the settings of a user in **user** > **Restrictions** > **Password Restrictions** as shown in the image.

Figure 26-5



OpenText eDirectory 9.2.7 and onwards, the settings can still be changed. But as soon as the user logs in, these settings are overwritten by the password policy. This does not only effect password expiration but also `passwordExpirationInterval` (**Force periodic password changes**), **Allow user to change password**, **Require unique passwords**. This has the effect that all settings in that regard that were done on user level with assigning a policy are now overwritten by the policy.

If **Universal Password** is disabled, this option will also be disabled by default. You can still enable **Universal Password** manually in the Password Policy window.

Figure 26-6 *Policy Summary*

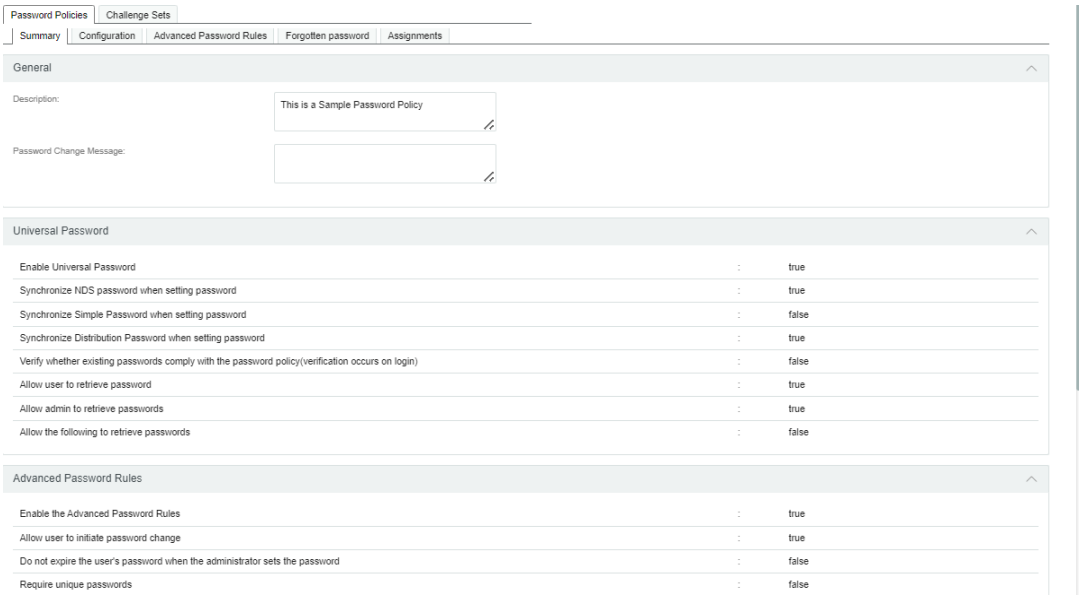
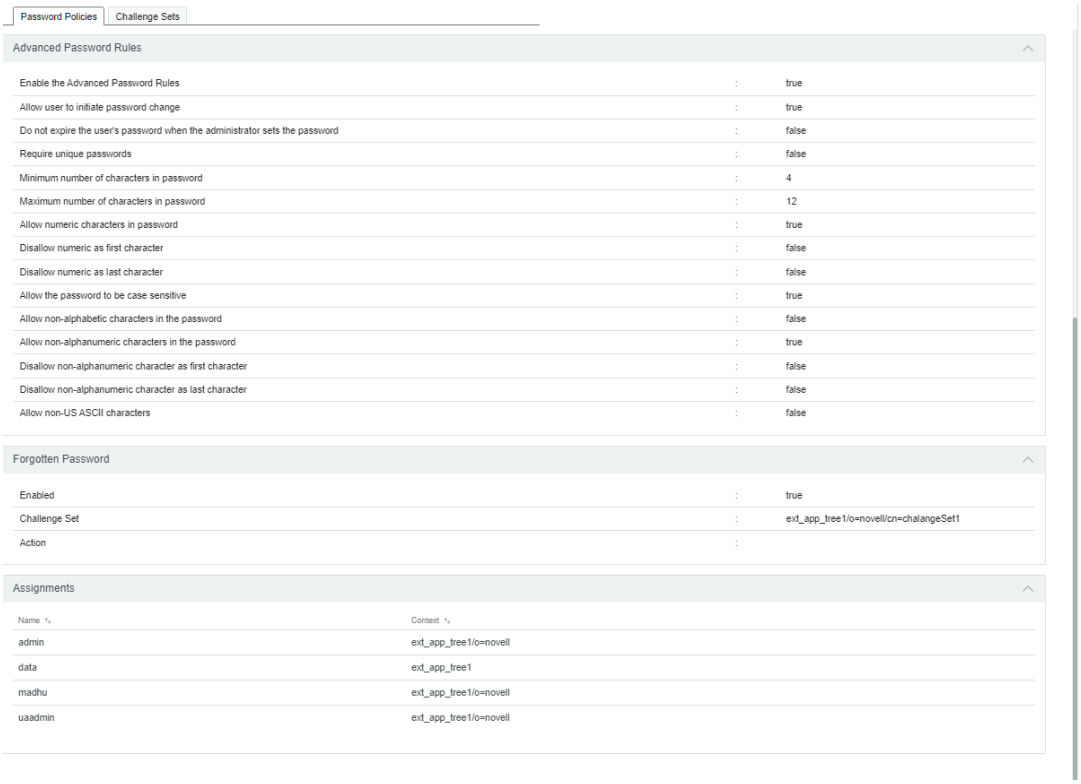


Figure 26-7 *Policy Summary Continued*



NOTE: Non-admin users can change their password only if the administrator enable the **Allow user to change password** and **Require a password** check boxes in **user** > **Restrictions** > **Password Restrictions** window. However, any settings done on the user object (assigned with a password policy) will be overwritten by the policy rules.

Assigning Password Policies to Users

You can assign a password policy to users in OpenText eDirectory by assigning the policy to the whole tree by using the Login Policy object, to specific partitions or containers, or to specific users. We encourage you to set password policies as high up in the tree as you can, to simplify administration.

IMPORTANT: Assigning a password policy to an entire OpenText eDirectory tree or to a container in a tree that contains a very large number of users (tens of thousands) in subcontainers can cause OpenText Identity Console to hang.

In this case, you might want to consider individually assigning password policies to lower-level containers in order to control the number of users for each password policy assignment.

A policy is not in effect until you assign it to one or more objects. You can assign a password policy to the following objects:

- Login Policy object

  We recommend that you create a default password policy for all users in the tree. You do this by creating a policy and assigning it to the Login Policy object. The Login Policy object is located in the Security container just below the root of the tree.

- A container that is a partition root

  If you assign a policy to a container that is the root of a partition, the policy assignment is inherited by all users in that partition, including users in subcontainers. To determine whether a container is a partition root, browse for the container and note whether a partition icon is displayed beside it.

- A container that is not a partition root

  If you assign a policy to a container that is not the root of a partition, the policy assignment is inherited only by users in that specific container. It is not inherited by users that are in subcontainers. If you want the policy to apply to all users below a container that is not a partition root, you must assign the policy to each subcontainer individually.

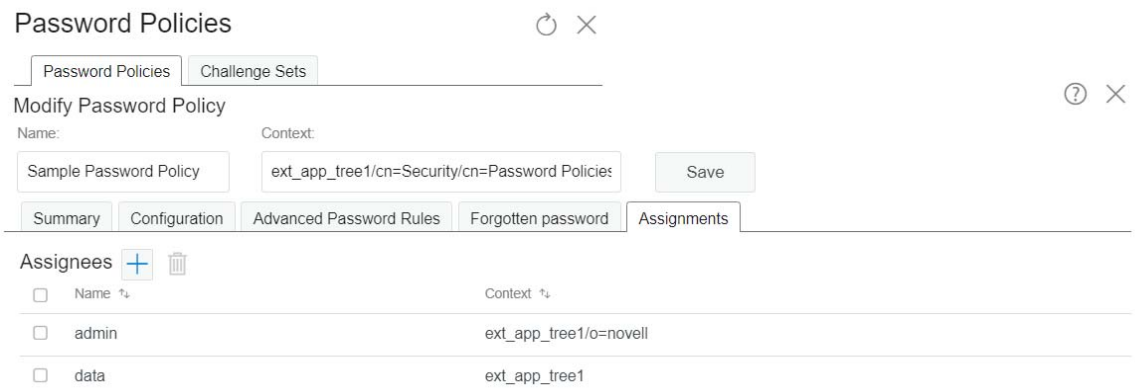
- A specific user

Only one policy is effective for a user at a time. NMAS determines which policy is effective for a user by looking for policies in the following order and applying the first one it finds.

1. **Specific user assignment:** If a password policy has been assigned specifically to the user, that policy is applied.
2. **Container:** If the user has no specific assignment, NMAS applies the policy that is assigned to the container that holds the user.
3. **Partition root container:** If no policy is assigned to the user or to the container directly above the user, the policy assigned to the partition root container is applied.
4. **Login Policy object:** If no policy is assigned to the user or other containers, the policy assigned to the Login Policy object is applied. It is the default policy for all users in the tree.

The following figure shows an example of the property page where you specify which object password policy is assigned to:

Figure 26-8 *Password Policy Assignments*



Finding Out Which Policy a User Has

Only one policy is in effect for a user at a time. To find out which policy is in effect for a particular user or container:

1. On the OpenText Identity Console home page, click **Authentication Management** tile > **Password Policies**.
2. Click on any Password policy and navigate to **Assignments** tab.

If there are multiple policies in the tree, NMAS determines which policy to apply to a user as described in "Assigning Password Policies to Users" on page 739.

Setting A User's Password

Administrators or help desk personnel can set a user's Universal Password by using a task in OpenText Identity Console. The task shows the password rules for the password policy that is in effect for the user.

1. On the OpenText Identity Console home page > **Authentication Management** > **Password Policies**.
2. Click **Create Password policy** +, and create a password policy with **Enable Universal Password** selected.
3. On the OpenText Identity Console home page, click **User Management** tile.
4. Browse to and select the desired user.

   If the user has a password policy assigned and Universal Password enabled, you can change the password.

   If the Advanced Password Rules are enabled in the policy, you see a list of rules under **Set Password** that must be followed.
5. Create a password for the user, making sure it is compliant with all password rules that are displayed.
6. Click **Save**.

The Universal Password is changed for the user.

If Password Synchronization is set up in your environment, the user's new password is distributed to the connected systems that are configured to accept it.

NOTE: If an administrator changes a user's password, such as when creating a new user or in response to a help desk call, the password is automatically expired if you have enabled the setting to expire passwords in the password policy. The setting, named **Number of days before password expires**, is in Advanced Password Rules. For this particular feature, the number of days is not important, but the setting must be enabled.

The **Do not expire the user's password when the administrator sets the password** option overrides this feature.

Universal Password Diagnostic Utility

OpenText eDirectory provides an utility which checks the status and re-encrypts the Universal Password. The Universal Password Diagnostic Utility (`diagpwd`) is a tool that allows an administrator to view the status of user's Universal Password (UP), Simple password, NDS password and Distribution Password (DP). It also reports the synchronization status of these passwords. Below is a sample syntax of the `diagpwd` utility:

```
diagpwd LDAP_SERVER_ADDR TLS_PORT CA_CERT_FILE SEARCH_BASE SEARCH_SCOPE
BIND_DN [BIND_PWD] -t
```

Option	Description
`LDAP_SERVER_ADDR`	Specifies address of the target LDAP server.
`TLS_PORT`	Specifies the LDAP Secure port (TLS) of the target LDAP server.
`CA_CERT_FILE`	Specifies path of the PEM encoded file containing the Trusted Root Certificate for the target LDAP server.
`SEARCH_BASE`	Use searchbase as the starting point for the search.
`SEARCH_SCOPE`	Specifies the scope of the search. Scope should be base, one, or sub to specify a base object, one-level, or subtree search.
`BIND_DN`	The LDAP DN of the administrator. For example: `cn=admin,o=company`.
`BIND_PWD`	The LDAP password of the administrator. **NOTE:** This parameter is optional. If it is not included in the command line, the user will be prompted for it.

Option	Description
`-t`	This re-encrypts UP, DP, simple password and password history with 256-bit AES keys. Use this option after creating AES 256-bit tree key. **NOTE:** While using this option, ensure that the Password policy allows the user running this utility to retrieve the user's UP.

NOTE: `diagpwd` utility is supported with OpenText eDirectory 9.1 SP2 and above.

Installing the diagpwd Utility

To install the `diagpwd` utility, run the following command in the OpenText eDirectory Setup folder:

```
rpm -i /eDirectory-setup-location/novell-nmas-ldap-ext-client-9.2.0-0.x86_64.rpm
```

Examples

To examine the status of passwords for user `cn=user1,ou=users,o=company` on server `192.168.1.1`, run the following command:

```
diagpwd 192.168.1.1 636 /home/user1/cert.pem cn=user1,ou=users,o=company base cn=admin,o=company
```

To examine the status of passwords for all users under `ou=users,o=company` subtree, run the following command:

```
diagpwd 192.168.1.1 636 /home/user1/cert.pem ou=users,o=company sub cn=admin,o=company
```

To re-encrypt passwords of all users under `ou=users,o=company` subtree with AES 256-bit key, run the following command:

```
diagpwd 192.168.1.1 636 /home/user1/cert.pem ou=users,o=company sub cn=admin,o=company -t
```

Troubleshooting Password Policies

- "Errors Indicate a Password Policy Is Not Assigned to a User" on page 743
- "Using Challenge Response Questions" on page 743
- "Giving Access to Users in New Containers" on page 743
- "NMAS LDAP Transport Error" on page 743

Errors Indicate a Password Policy Is Not Assigned to a User

If you see an error saying that a password policy is not assigned to a user from the Set Universal Password task, and you know that the user does have a password policy assigned, SSL might be the issue. To diagnose and resolve SSL issues, perform the following tasks:

- To help confirm that SSL configuration is the problem, use the View Policy Assignment task to check the policy for that user. If the View Policy Assignment task displays an NMAS Transport error, this can be an indicator that SSL is not configured properly.
- Make sure that SSL is configured correctly between the Web server running OpenText Identity Console and the primary OpenText eDirectory tree. Confirm that you have a certificate configured between the Web server and OpenText eDirectory.
- If you are not requiring TLS for simple bind, you must make sure you indicate the correct LDAP SSL port, as explained in the note in Step 3 on page 721.

Using Challenge Response Questions

Make sure that you are using a supported browser for OpenText Identity Console.

Giving Access to Users in New Containers

When you set up OpenText Identity Console or one of OpenText portal products, such as User Application, you specify the portal users container. Usually you specify a container at a high level in the tree, so that all users in the tree can access portal features. If all your users are below that container, then all users have access to Forgotten Password and Reset Password Self-Service.

NMAS LDAP Transport Error

If you are installing Identity Manager in a multiserver environment and use some of the Password Management feature in OpenText Identity Console, you might see an error that begins with `NMAS LDAP Transport Error`.

One common cause of this error is that the `PortalServlet.properties` file is pointing to an LDAP server that does not have the NMAS extensions that are needed for Identity Manager. Open the `PortalServlet.properties` file and make sure the address for the LDAP server is the same server where you installed Identity Manager.

Other possible causes:

- The LDAP server is not running.
- SSL is not configured for LDAP between the OpenText Identity Console server and the LDAP server.
- When logging in to other trees with OpenText Identity Console to manage remote Identity Manager servers, you might encounter errors if you use the server name instead of the IP address for the remote server.
- The trusted root certificate of the tree you authenticate to must be imported as a trusted certificate onto the Web server. You can use `keytool.exe` to export the certificate to the Web server.

Security Considerations

Reversible encryption of Universal Password is required for convenient interoperation with other password systems. Administrators must evaluate the costs and benefits of the system. Using a Universal Password stored in OpenText eDirectory might be more secure or convenient than attempting to manage several passwords.

A Universal Password in OpenText eDirectory is protected by three levels of security: triple DES encryption of the password itself, OpenText eDirectory rights, and file system rights.

- Prior to NICI 3.0, the Universal Password was encrypted by a triple DES, user-specific key. Both the Universal Password and the user key were stored in system attributes that only OpenText eDirectory can read. The user key (3DES) was stored encrypted with the tree key, and the tree key was protected by a unique NICI key on each machine. Note that neither the tree key nor the NICI key was stored within OpenText eDirectory. They were not stored with the data they protect. The tree key was present on each machine within a tree, but each tree had a different tree key, so data encrypted with the tree key could be recovered only on a machine within the same tree. Thus, while stored, the Universal Password was protected by three layers of encryption.

  NICI 3.0 supports AES 256-bit storage keys; therefore, any application that uses the storage keys to securely wrap other keys should be able to handle the new algorithm. However, any data which is currently wrapped with the older 3-DES keys will still be assessable without any changes.

  NICI 3.0 supports AES 256-bit tree key. However, OpenText eDirectory does not create the AES 256-bit tree key by default. Creating this key in a an environment with 9.0 and earlier versions can cause issues in services that depend on the tree key. You are recommended to update all your OpenText eDirectory servers to 9.2 before creating the key. For more information, see Creating an AES 256-Bit Tree Key.
- Each key is also secured via OpenText eDirectory rights. Only administrators with the Supervisor right or the users themselves have the rights to change Universal Passwords.

NOTE: The password policy can be configured to allow Universal Password to be read by administrators and for users to read their own passwords through using NMAS/nds-cluster-config extensions. This is not enabled by default.

- File system rights ensure that only a user with the proper rights can access keys.

  If Universal Password is deployed in an environment requiring high security, you can take the following additional precautions:

  - Make sure that the following directories and files are secure:

Windows	%SystemRoot%\SysWOW64\Novell\nici %SystemRoot%\System32\ where the NICI DLL is installed
Linux	/var/opt/novell/nici /etc/opt/novell/nici64.cfg /opt/novell/lib64/libccs2.so and the NICI shared libraries in the same directory

Consult the documentation for your system for specific details of the location of NICI and OpenText eDirectory files.

- As with any security system, restricting physical access to the server where the keys reside is very important.

For security consideration relating to password management, see "Security Considerations" on page 629.

Importing Hash Based Passwords Into eDirectory

Passwords can be imported into eDirectory via an LDIF in DIGEST-MD5, crypt, SHA, and SSHA hash. Perform the following steps to import the MD5 hash based passwords into OpenText eDirectory:

1. Create a MD5 hash in base64 format using the following command:

   ```
   echo -n <password> | openssl md5 -binary | base64
   ```

   NOTE: eDirectory supports hash based passwords only in base64 format.

2. Add the text which is returned while creating the MD5 hash in an LDIF file as shown in the below example:

   ```
   dn: cn=sp1,o=novell
   control: 2.16.840.1.113719.1.27.101.5
   changetype: modify
   replace: userPassword
   userPassword: {md5}CSbJUP4kfDtGXrE+JY7kaNI5oGU=
   ```

   NOTE: Ensure that there is no password policy applied to the user that is modified via the LDIF file.

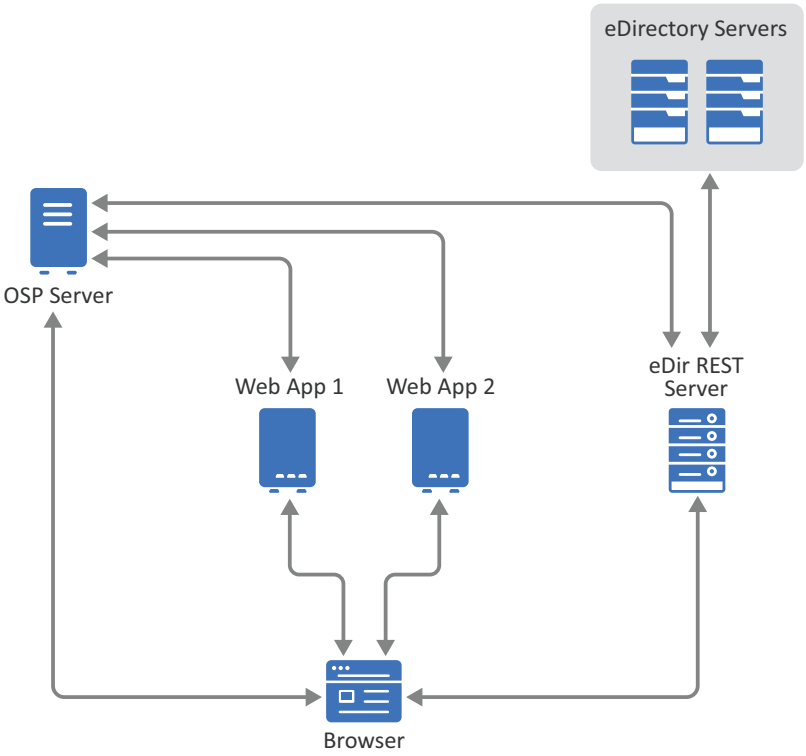
3. Install both Simple Password and DIGEST-MD5 NMAS methods and make the Simple Password method as the default method.
4. Use ice with the -l option for the LDAP destination handler using the following command:

   ```
   ice -S LDIF -f ./change_pass.ldiff -D LDAP -s 164.99.163.236 -p 636 -d
   cn=admin,o=novell -w n -l -L /var/opt/novell/eDirectory/data/SSCert.der
   ```

27 REST Services

OpenText eDirectory incorporates several REST APIs that enable various features of OpenText eDirectory to be accessed via REST web services. Using the APIs, you can invoke any request to get the respective response from the OpenText eDirectory server. REST web services can be deployed as Docker container. For more information, refer to the below diagram depicting the architecture of the REST web services for OpenText eDirectory:

Figure 27-1 *Architecture of REST Web Services with OpenText eDirectory*



NOTE: OpenText eDirectory REST services use OAUTH2 protocol to provide authentication with One SSO Provider (OSP).

This document tells you how to perform the following tasks:

- "Security Recommendations" on page 748
- "Planning to Install REST Services for OpenText eDirectory" on page 748
- "Configuring REST Services for OpenText eDirectory" on page 751
- "Managing Data Persistence" on page 753
- "Auditing with REST Services" on page 754
- "Modifying LDAP Password Using REST Container" on page 755
- "Modifying Server Certificate Using REST Container" on page 755

- "Upgrading REST Services for OpenText eDirectory" on page 756
- "REST API Documentation" on page 757

Security Recommendations

- Docker containers do not have any resource constraints by default. This provides every container with the access to all the CPU and memory resources provided by the host's kernel. You must also ensure that one running container should not consume more resources and starve other running containers by setting limits to the amount of resources that can be used by a container.
  - Docker container should ensure that a Hard Limit is applied for the memory used by the container using the `--memory` flag on Docker run command.
  - Docker container should ensure that a limit is applied to the amount of CPU used by a running container using the `--cpuset-cpus` flag on the Docker run command.
- `--pids-limit` should be set to 300 to restrict the number of kernel threads spawned inside the container at any given time. This is to prevent DoS attacks.
- You must set the on-failure container restart policy to 5 using the `--restart` flag on Docker run command.
- You must only use the REST container once the health status shows as **Healthy** after the container comes up. To check the container's health status, run the following command:

  `docker ps <container_name/ID>`
- REST container will always start as non-root user (`nds`). As an additional security measure, enable user namespace remapping on the daemon to prevent privilege-escalation attacks from within the container. For more information on user namespace remapping, see Isolate containers with a user namespace.

Planning to Install REST Services for OpenText eDirectory

This section explains how to prepare your setup before installing the REST services. In order to install and configure REST, you must perform the following tasks:

- Ensure to get a pkcs12 server certificate. You can use server certificates generated by any external CA or OpenText Identity Console. For more information, see "Creating a Server Certificate Object" on page 647.
- Ensure to get a CA certificate file in `.pem` format. For example, you can use the OpenText eDirectory CA certificate (SSCert`.pem`).
- (Optional) Install and configure OSP before installing REST services. For more information, see Deploying OSP Container.
- Create a configuration file with the following configuration parameters. For example, create `edirapi.conf` file. The values for the configuration file can be changed according to your requirement.

  NOTE: Two sample configuration files will be bundled with the tarball image. You can choose to make changes to those files according to your business requirement.

In case, you want to configure REST with OSP, create a configuration file as shown below:

```
listen = ":9000"
ldapserver = "192.168.1.1:636"
ldapuser = "cn=admin,ou=sa,o=system"
ldappassword = "novell"
pfxpassword = "novell"
ospmode = "true"
osp-token-endpoint = "https://10.10.10.10:8543/osp/a/idm/auth/oauth2/
getattributes"
osp-authorize-url = "https://10.10.10.10:8543/osp/a/idm/auth/oauth2/
grant"
osp-logout-url = "https://10.10.10.10:8543/osp/a/idm/auth/app/logout"
osp-redirect-url = "https://10.10.10.10:9000/eDirAPI/v1/edirtree/
authcoderedirect"
osp-client-id = "identityconsole"
ospclientpass = "novell"
ospcert = "/etc/opt/novell/eDirAPI/cert/SSCert.pem"
bcert = "/etc/opt/novell/eDirAPI/cert/"
loglevel = "error"
check-origin = "true"
origin = "https://10.10.10.10:9000,https://192.168.1.1:8543"
```

In case, you want to configure REST without OSP, create a configuration file as shown below, without the OSP parameters:

```
listen = ":9000"
pfxpassword = "novell"
ospmode = "false"
bcert = "/etc/opt/novell/eDirAPI/cert/"
edir-hosts = "<ip_address-1>:636,<ip_address-2>:636"
```

Table 27-1 *Description of the configuration parameters in the configuration file*

Configuration Parameters	Description
listen	Specify 9000 as the REST server's listener port inside the container.
ldapserver	Specify the OpenText eDirectory host server IP
ldapuser	Specify the username of the user with admin rights to the eDirectory tree. **IMPORTANT:** The username must be in lowercase. The eDirAPI docker container may fail to initialize if the username is in uppercase.
ldappassword	Specify the password of the LDAP server
pfxpassword	Specify the password of the .pfx certificate file
ospmode	Specify true to integrate OSP with OpenText Identity Console. If you set this to false, OpenText Identity Console will use ldap login

Configuration Parameters	Description
osp-token-endpoint	This URL is used to fetch certain attributes from the OSP server to verify the validity of the authentication token
osp-authorize-url	This URL is used by the user to provide credentials to obtain an authentication token
osp-logout-url	Use this URL to terminate the session between the user and the OSP server
osp-redirect-url	The OSP server re-directs the user to this URL after granting the authentication token
osp-client-id	Specify the OSP client ID which was provided at the time of the REST registration with OSP
ospclientpass	Specify the OSP client password which was provided at the time of the REST registration with OSP
ospcert	Specify the location of OSP server's CA certificate
bcert	Specify location of OpenText Identity Console"s CA certificate
loglevel	Specify the log levels that you want to include in the log file. This parameter can be set to "fatal", "error", "warn" or "info".
check-origin	If this is set to true, the REST server compares the origin value of requests. Available options are either true or false. The *origin* parameter is mandatory even if *check-origin* parameter value is set to false when DNS configuration is used.
origin	eDirAPI compares the origin value of requests with the values specified in this field. **NOTE:** From eDirAPI 1.4 onward, this parameter is independent of check-origin parameter and is mandatory if DNS configuration is used.
enableaudit	Set this option to true to enable auditing for REST services. Available options are either true or false.
enableservicestartaudit	Set this option to true to get notified for REST service start events. Available options are either true or false.
enableservicestopaudit	Set this option to true to get notified for REST service stop events. Available options are either true or false.
enablelogsessioncreationaudit	Set this option to true to get notified for REST service session creation events. Available options are either true or false.
enablelogsessionterminationaudit	Set this option to true to get notified for REST service session termination events. Available options are either true or false.

Configuration Parameters	Description
auditlogmaxsize	Specify the maximum limit of each REST service's audit log file size. By default, the file size is 50 MB.
edirapilogmaxsize	Specify the maximum limit of each REST server's log file size.
scope	Specify the scope of REST server when it is used as a resource server in OAuth terminology. By default, it is set to edirapi *<tree_name>*.
maxclients	Maximum number of concurrent clients which can access `edirapi`. Any additional clients beyond this limit have to wait in queue.
edir-hosts	edir-host is a parameter that contains the IP address(s) of eDirectory which you want to connect through OpenText Identity Console.

IMPORTANT: ◆The OSP related configuration parameters should be used only if you plan to integrate OSP along with REST services.

- To enable auditing for REST services, you must configure the auditing related parameters in the configuration file.
- OSP HTTPS URL should be validated with certificates containing 2048 bit key. This validation fails with certificates that contain 4096 or 8192 bit keys.

Configuring REST Services for OpenText eDirectory

Perform the following steps to configure REST APIs for OpenText eDirectory:

1. Download `eDirAPI_<version>_Container.tar.gz` from Software License and Download portal. For example, `eDirAPI_140_Container.tar.gz`.
2. The image has to be loaded into the local Docker registry by using the following commands:

   ```
   tar -xvf eDirAPI_140_Container.tar.gz
   ```

   ```
   docker load --input eDirAPI_140/eDirAPI_140.tar.gz
   ```

3. Create a docker container using the following command:

   ```
   docker create --name edirapi-container --volume <volume-name>:/config/
   --network=<network-type> --env ACCEPT_EULA=Y edirapi:<version>
   ```

   For example,

   ```
   docker create --name edirapi-container --volume edirapi-volume:/config/
   --network=host --env ACCEPT_EULA=Y edirapi:1.4.0.0000
   ```

NOTE: ◆You can accept the EULA by setting ACCEPT_EULA environment variable to Y. You can also accept the EULA from the on-screen prompt while starting the container by using `-it` option in the Docker create command for interactive mode.

- `--volume` parameter in the above command will create a volume for storing configuration and log data. In this case, we have created a sample volume called edirapi-volume.

4 Copy the server certificate file (`.pfx`) from your local file system to the container in `/etc/opt/novell/eDirAPI/cert/keys.pfx` using the following command:

```
docker cp <absolute path of server certificate file> edirapi-container:/etc/opt/novell/eDirAPI/cert/keys.pfx
```

For example,

```
docker cp /home/user/keys.pfx edirapi-container:/etc/opt/novell/eDirAPI/cert/keys.pfx
```

5 Copy the CA certificate file (`.pem`) from your local file system to the container in `/etc/opt/novell/eDirAPI/cert/SSCert.pem` using the following command:

```
docker cp <absolute path of CA certificate file> edirapi-container:/etc/opt/novell/eDirAPI/cert/SSCert.pem
```

For example,

```
docker cp /home/user/SSCert.pem edirapi-container:/etc/opt/novell/eDirAPI/cert/SSCert.pem
```

When you connect to multiple OpenText eDirectory trees, you must ensure to obtain individual CA certificate for all the connected trees. For example, if you connect to three OpenText eDirectory trees, then you must copy all the three CA certificates in to docker container:

```
docker cp /home/user/SSCert.pem edirapi-container:/etc/opt/novell/eDirAPI/cert/SSCert.pem
docker cp /home/user/SSCert1.pem edirapi-container:/etc/opt/novell/eDirAPI/cert/SSCert1.pem
docker cp /home/user/SSCert2.pem edirapi-container:/etc/opt/novell/eDirAPI/cert/SSCert2.pem
```

6 Copy the configuration file (`edirapi.conf`) from your local file system to the container in `/etc/opt/novell/eDirAPI/conf/edirapi.conf` using the following command:

```
docker cp <absolute path of CA certificate file> edirapi-container:/etc/opt/novell/eDirAPI/conf/edirapi.conf
```

For example,

```
docker cp /home/user/edirapi.conf edirapi-container:/etc/opt/novell/eDirAPI/conf/edirapi.conf
```

NOTE: To deploy eDirAPI with disable anonymous bind, you must have “ldapserver”, “ldapuser”, and “ldappassword” parameters in edirapi.conf file.

A sample configuration file is shown below:

```
listen = ":9000"
pfxpassword = "novell"
bcert = "/etc/opt/novell/eDirAPI/cert/"
ospmode=false
edir-hosts = "<ip_address>:636"
```

NOTE: To access the OpenText eDirectory through OpenText Identity Console, it is required to add the edir-hosts="x.x.x.x:ldaps_port in the edirapi.conf file.

For example,

edir-hosts="10.10.10.10:636"

7 Start the Docker container using the following command:

```
docker start <edirapi-container-name>
```

For example,

```
docker start edirapi-container
```

The default log file location inside the REST container will be `/config/eDirAPI/var/log/`. You will find the following log files in this location:

- `container-startup.log`
- `edirapi.log`

NOTE:
- The CA certificate for your REST server should begin with `BEGIN CERTIFICATE` and end with `END CERTIFICATE`. If you provide any other value, REST server will display an error message.
- To support connections up to 42 thousand in your REST container, you must increase the port range by running the following three commands:

```
ulimit -n 999999
cat /proc/sys/net/ipv4/ip_local_port_range
echo 1024 65535 > /proc/sys/net/ipv4/ip_local_port_range
```

- You can ignore the following message in the container startup log file:

```
Setting IDCONSOLEMODE from Environment to false
```

Managing Data Persistence

Along with the REST containers, volumes for data persistence are also created. To use the configuration parameters of an old container using the volumes, perform the following steps:

1 (Optional) Stop your current docker container using the following command:

```
docker stop edirapi-container
```

2 Create the second container using the following command:

```
docker create --name edirapi-container-2 --network=host --volume
edirapi-volume-2:/config/ edirapi:1.4.0<version>
```

3 Start the second container using the following command:

```
docker start edirapi-container-2
```

4 (Optional) Now the first container can be removed using the following command:

```
docker rm edirapi-container
```

Auditing with REST Services

OpenText eDirectory REST server is capable of sending CEF auditing event logs to the Sentinel server. REST server sends the audit data using ArcSight Smart Connectors. For more information on how to configure and use SmartConnectors, see ArcSight SmartConnector User Guide.

To enable auditing for REST services, you must configure the auditing related parameters in the configuration file. For more information, see “Planning to Install REST Services for OpenText eDirectory” on page 748.

Understanding REST Events

By default, all REST events are enabled. You can disable any specific event in case it is not required for your Organization. OpenText eDirectory REST server is capable of auditing the following events:

Event	Description
ENABLESERVICESTARTAUDIT	This generates an event in case of starting the REST service
ENABLESERVICESTOPAUDIT	This generates an event in case of stopping the REST service
ENABLELOGSESSIONCREATIONAUDIT	This generates an event when a REST session is created
ENABLELOGSESSIONTERMINATIONAUDIT	This generates an event when a REST session is terminated

NOTE: The logs are being found `/var/opt/novell/eDirAPI/log/edirapi_auditlog.log`.

Example

Find the following example of **Create Session** event:

```
Oct 10 15:37:17 eDirAPI
CEF:0|NetIQ|eDirAPI|1.0|000B0510|SESSION_CREATE|3|dvc=10.71.128.233
dvchost=SLES12SP3-SHREYAS-128233 rt=Oct 10 2019 15:37:17 dtz=IST
src=164.99.136.60 spt=59132 suser=cn\=admin,o\=novell
duser=cn\=admin,o\=novell cn1Label=CorrelationID cn1=rtpL9xt-
tzBR92fEGt9rrczA_1M2vHrGM4Q_8AjEmSU= cs1Label=Client Address
cs1=164.99.136.60 cs2Label=Tree Name cs2=SHREYAS_TREE2 sproc=eDirAPI
sourceServiceName=edirapi reason=201 outcome=Success
```

Modifying LDAP Password Using REST Container

Perform the following steps to modify LDAP Passwords using REST container:

1. Login to your container using the following command:

   ```
   docker exec -it <container_name> bash
   ```

2. Store a new password in the password store using the following command:

   ```
   LD_LIBRARY_PATH=/opt/novell/lib64/:/opt/novell/eDirectory/lib64/:/opt/
   netiq/common/openssl/lib64/ /opt/novell/eDirAPI/sbin/passwdstore -a
   <Admin DN>
   ```

   The above command will prompt for a password. Enter the new password.

   For example,

   ```
   LD_LIBRARY_PATH=/opt/novell/lib64/:/opt/novell/eDirectory/lib64/:/opt/
   netiq/common/openssl/lib64/ /opt/novell/eDirAPI/sbin/passwdstore -a
   cn=admin,o=novell
   ```

3. Exit the container console using the following command:

   ```
   exit
   ```

4. Restart the container

   ```
   docker restart <container name>
   ```

Modifying Server Certificate Using REST Container

Perform the following steps to modify server certificates using REST container:

1. Run the following command to copy the new server certificate (for e.g. `new-keys.pfx`) in any location of your container:

   ```
   docker cp /path/to/new-keys.pfx <container_id/name>:/tmp/new-keys.pfx
   ```

2. Login to your container using the following command:

   ```
   docker exec -it <container_name> bash
   ```

3. Run `NLPCERT` to store the keys.

   ```
   LD_LIBRARY_PATH=/opt/novell/lib64/:/opt/novell/eDirectory/lib64/:/opt/
   netiq/common/openssl/lib64/ /opt/novell/eDirAPI/sbin/nlpcert -i /tmp/
   new-keys.pfx -o /etc/opt/novell/eDirAPI/conf/ssl/private/cert.pem
   ```

   The above command will also prompt you to enter the server certificate password. Enter your password.

4. Exit the container console using the following command:

   ```
   exit
   ```

5. Restart the container

   ```
   docker restart <container name>
   ```

Upgrading REST Services for OpenText eDirectory

When a new version of REST Image is available, the administrator can perform an upgrade procedure to deploy container with the latest version of REST services. Ensure to store all necessary application related data persistently in Docker volumes before performing an upgrade. Perform the following steps to upgrade REST services using Docker container:

1. Download and load the latest version of the Docker image from the Software License and Download portal and perform the steps to install the latest version of OpenText eDirectory.
2. Once the latest docker image is loaded, stop your current docker container using the following command:

   ```
   docker stop <edirapi-container-name>
   ```

   For example,

   ```
   docker stop edirapi-container-1
   ```

3. Delete the existing OpenText eDirectory container by running the following command:

   ```
   docker rm < edirapi -container-name>
   ```

   For example,

   ```
   docker rm edirapi -container-1
   ```

4. (Optional) Delete the obsolete OpenText eDirectory Docker image by running the following command:

   ```
   docker rmi <edirapi-image ID>
   ```

   For example,

   ```
   docker rmi edirapi: 1.4.1.0000
   ```

5. Create a new container using the new REST Docker image using the following command:

   ```
   docker create --name < edirapi -container-name> --env ACCEPT_EULA=Y --
   network=<network-type> --volume <volume-name>:/config/
   edirapi:<version>
   ```

   For example,

   ```
   docker create --name edirapi -container-2 --env ACCEPT_EULA=Y --
   network=host --volume edirapi -volume:/config/ edirapi: 1.4.1.0000
   ```

6. Copy the configuration file (`edirapi.conf`) from your local file system to the newly created container as `/etc/opt/novell/eDirAPI/conf/edirapi.conf` using the following command:

   ```
   docker cp <absolute path of configuration file> edirapi-
   containername:/etc/opt/novell/eDirAPI/conf/edirapi.conf
   ```

   For example,

```
docker cp /home/user/edirapi.conf edirapi-container-2:/etc/opt/novell/
eDirAPI/conf/edirapi.conf
```

A sample configuration file is shown below:

```
listen = ":9000"

pfxpassword = "novell"

bcert = "/etc/opt/novell/eDirAPI/cert/"

ospmode=false

edir-hosts = "<ip_address-1>:636,<ip_address-2>:636"
```

NOTE:

- While upgrading to OpenText eDirectory 1.7.2 and above, it is required to add OpenText eDirectory server IP to `edirapi.conf` file before copying it to the container.
- If you want OpenText eDirectory to connect to multiple OpenText eDirectory trees, enter their IP addresses or domain names separated by commas.

7 Start the second container using the following command:

```
docker start <edirapi-container-name>
```

For example,

```
docker start edirapi -container-2
```

8 To check status of the running container, run the following command:

```
docker ps -a
```

NOTE: REST Services 1.4 is the next release after REST 1.3. There was no interim release for REST services in between these two releases. Post 1.4, OpenText eDirectory released REST 1.4.0.0100 which adds support for OpenSSL 1.0.2zd. This OpenSSL version includes a fix for a potential security vulnerability. For more information about the issue, see this page.

REST API Documentation

REST API documentation can be accessed using one of the following links:

- API documentation for Resource Server Mode (https://www.netiq.com/documentation/edirectory-93/resources/ResourceModeOpenapi/). In this mode, authentication happens using OSP token.
- API documentation for Simple Login Mode (https://www.netiq.com/documentation/edirectory-93/resources/SimpleLoginOpenapi/). In this mode, authentication happens using the LDAP username and password.

A NMAS Considerations

This appendix contains the following topics:

- "Setting Up a Security Container As a Separate Partition" on page 759
- "Merging Trees with Multiple Security Containers" on page 759

Setting Up a Security Container As a Separate Partition

NMAS relies on the storage of policies that are global to the OpenText eDirectory tree. The OpenText eDirectory tree is effectively the security domain. The security policies must be available to all servers in the tree.

NMAS places the authentication policies and login method configuration data in the Security container that is created off of the [Root] in OpenText eDirectory trees. This information must be readily accessible to all servers that are enabled for NMAS. The purpose of the Security container is to hold global policies that relate to security properties such as login, authentication, and key management.

With NMAS, we recommend that you create the Security container as a separate partition, and that the container be widely replicated. This partition should be replicated as a Read/Write partition only on those servers in your tree that are highly trusted.

NOTE: Because the Security container contains global policies, be careful where writable replicas are placed, because these servers can modify the overall security policies specified in the OpenText eDirectory tree. In order for users to log in with NMAS, replicas of the User objects must be on the NMAS server.

Merging Trees with Multiple Security Containers

Special considerations need to be made when merging OpenText eDirectory trees where a Security container has been installed in one or both of the trees. Make sure that this is something you really want to do because this procedure has the potential to be a very time-consuming and laborious task.

To merge trees with multiple Security containers:

1. In OpenText Identity Console, identify the trees that will be merged.
2. Identify which tree will be the source tree and which tree will be the target tree.

   Keep in mind these security considerations for the source and target trees:

   - Any certificates signed by the source tree's Organizational CA must be deleted.
   - The source tree's Organizational CA must be deleted.
   - All user secrets stored in OpenText SecretStore on the source tree must be deleted.

- All NMAS login methods in the source tree must be deleted and reinstalled in the target tree.
- All NMAS users that were in the source tree must be re-enrolled when the trees are merged.
- All users and servers that were in the source tree must have new certificates created for them when the trees are merged.
- All users that were in the source tree must have their secrets reinstalled into SecretStore.

If neither the source tree nor the target tree has a container named Security under the root of the tree, or if only one of the trees has the Security container, no further action is required. Otherwise, continue with the remaining procedures in this section.

NOTE: Two OpenText eDirectory trees which have EBA enabled servers should not be merged.

Product-Specific Operations to Perform prior to Tree Merge

This section contains the following information:

- "OpenText Certificate Server" on page 760
- "OpenText Single Sign-on" on page 761
- "NMAS" on page 762
- "OpenText Security Domain Infrastructure" on page 762
- "Other Security-Specific Operations" on page 763

OpenText Certificate Server

Depending on how the product was used, the objects and items referred to might or might not be present. If the objects and items referred to in a given step are not present in the source tree, you can skip the step.

1 Any Trusted Root certificates in the source tree should be installed in the target tree.

Trusted Root certificates are stored in Trusted Root objects, which are contained by Trusted Root containers. Trusted Root containers can be created anywhere within the tree. However, only the Trusted Root certificates that are in the Trusted Root containers within the Security container must be moved manually from the source tree to the target tree.

2 Install the Trusted Root certificates in the target tree.

2a Pick a Trusted Root container in the Security container in the source tree.

2b Create a Trusted Root container in the Security container of the target tree with the exact name used in the source tree (Step 2a).

2c In the source tree, open a Trusted Root object in the selected Trusted Root container and export the certificate.

IMPORTANT: Remember the location and filename you choose. You will use them in the next step.

2d In the target tree, create a Trusted Root object in the container that you created in Step 2b. Specify the same name as the source tree and, when prompted for the certificate, specify the file that you created in Step 2c.

2e Delete the Trusted Root object in the source tree.

2f Repeat Step 2c through Step 2e until all Trusted Root objects in the selected Trust Root container have been installed into the target tree.

2g Delete the Trusted Root container in the source tree.

2h Continue Step 2a through Step 2f until all Trusted Root containers have been deleted in the source tree.

3 Delete the Organizational CA in the source tree.

The Organizational CA object is in the Security container.

IMPORTANT: Any certificates signed by the Organizational CA of the source tree will become unusable following this step. This includes server certificates and user certificates that have been signed by the Organizational CA of the source tree.

---

4 Delete every Key Material object (KMO) in the source tree that has a certificate signed by the Organizational CA of the source tree.

Key Material objects in the source tree with certificates signed by other CAs will continue to be valid and do not need to be deleted.

If you are uncertain about the identity of the signing CA for any Key Material object, look at the Trusted Root Certificate section of the Certificates tab in the Key Material object property page.

5 Delete all user certificates in the source tree that have been signed by the Organizational CA of the source tree.

If users in the source tree have already exported their certificates and private keys, those exported certificates and keys will continue to be usable. Private keys and certificates that are still in OpenText eDirectory will no longer be usable after you perform Step 3.

For each user with certificates, open the properties of the User object. Under the Certificates section of the Security tab, a table lists all the certificates for the user. All of those certificates with the Organizational CA as the issuer must be deleted.

OpenText Single Sign-on

If OpenText Single Sign-on has been installed on any server in the source tree, you should delete all OpenText Single Sign-on secrets for users in the source tree.

For every user using OpenText Single Sign-on in the source tree, open the properties of the User object. All of the user's secrets will be listed under the SecretStore section of the Security tab. Delete all listed secrets.

NOTE: Depending on how the product was used, the objects and items referred to might or might not be present. If the objects and items referred to are not present in the source tree, you can skip this step.

NMAS

Depending on how the product was used, the objects and items referred to might or might not be present. If the objects and items referred to are not present in the source tree, you can skip the step.

1. In the target tree, install any NMAS login methods that were in the source tree but not in the target tree.

   To ensure that all of the necessary client and server login components are properly installed in the target tree, we recommend that you install all new login methods using original OpenText or vendor-supplied sources.

   Although methods *can* be reinstalled from existing server files, establishing a clean installation from OpenText or vendor-supplied packages is typically simpler and more reliable.

2. To ensure that the previously established login sequences in the source tree are available in the target tree, migrate the desired login sequences.

   2a In OpenText Identity Console, select the Security container in the source tree.

   2b Right-click the **Login Policy** object and select **Properties**.

   2c For each login sequence listed in the **Defined Login Sequences** drop-down list, note the Login Methods used (listed in the right pane).

   2d Select the Security container in the target tree and replicate the login sequences using the same login methods note in Step 2c.

   2e Click **OK** when you are finished.

3. Delete NMAS login security attributes in the source tree.

   3a In the Security container of the source tree, delete the Login Policy object.

   3b In the Authorized Login Methods container of the source tree, delete all login methods.

   3c Delete the Authorized Login Methods container in the source tree.

   3d In the Authorized Post-Login Methods container of the source tree, delete all login methods.

   3e Delete the Authorized Post-Login Methods container in the source tree.

   NOTE: To delete the Authorized Login Methods, use ldapdelete.

OpenText Security Domain Infrastructure

Depending on how the product was used, the objects and items referred to might or might not be present. If the objects and items referred to are not present in the source tree, you can skip the step.

1. Delete the W0 object and the KAP container in the source tree.

   The KAP container is in the Security container. The W0 object is in the KAP container.

2. On all servers in the source tree, delete the Security Domain Infrastructure (SDI) keys by deleting the `/var/opt/novell/nici/uid/nicisdi.key` file on Linux and %SystemRoot%\SysWOW64\Novell\NICI\nicisdi.key on WIndows.

   IMPORTANT: Make sure that you delete this file on *all* servers in the source tree.

Other Security-Specific Operations

If a Security container exists in the source tree, delete the Security container before you merge the trees.

Performing the Tree Merge

OpenText eDirectory trees are merged using the DSMerge utility. For more information, see Appendix B, “OpenText eDirectory Linux Commands and Usage,” on page 765 .

Product-Specific Operations to Perform after the Tree Merge

This section contains the following information:

- “OpenText Certificate Server” on page 763
- “OpenText Single Sign-On” on page 763
- “NMAS” on page 763

OpenText Certificate Server

If you are using OpenText Certificate Server, then after the tree merge reissue certificates for servers and users that were formerly in the source tree, as necessary.

OpenText Single Sign-On

If you are using OpenText Single Sign-on, after the tree merge you should re-create SecretStore secrets for users who were formerly in the source tree, as necessary.

NMAS

If you are using NMAS, after the tree merge you should re-enroll NMAS users who were formerly in the source tree, as necessary.

For more information, see Chapter 24, “Understanding OpenText eDirectory’s Authentication Framework,” on page 603.

B OpenText eDirectory Linux Commands and Usage

This chapter lists the utilities for OpenText eDirectory on Linux and their usage:

- "General Utilities" on page 765
- "LDAP-Specific Commands" on page 770

General Utilities

This section gives a list of the OpenText eDirectory utilities on Linux and their usage.

NOTE: After installation, ensure that you run the ndsconfig, ndscheck, and ndslogin utilities from the installed location on the server, which is `/opt/novell/eDirectory/bin` by default. Do not run ndsconfig from the installation package.

For more information on the usage of the OpenText eDirectory utilities, see the man page for each utility and "Troubleshooting Utilities on Linux" on page 834.

Command	Description	Usage
`nds-install`	Utility that installs OpenText eDirectory.	`nds-install [-h] [--help] [-i] [-j] [-u]`

Command	Description	Usage
ndsconfig	Configures OpenText eDirectory	ndsconfig <new> [-t <treename>] [-n <server context>] [-a <admin FDN>] [-w <password>] [-B ip_address1|interface1@port1,ip_address2|interface2@port2....] [-b port to bind] [-i] [-S <server name>] [-D <instance path>] [-d <path for dib>] [-m <module>] [-e] [-R|-r] [-c] [-L <ldap port>] [-l <SSL port>] [-P <LDAP URLs>] [-o http port] [-O https port] [--config-file <absolute path for configuration file>] [--configure-eba-now <yes/no>] ndsconfig <def> [-t <treename>] [-n <server context>] [-a <admin FDN>] [-w <password>] [-B ip_address1|interface1@port1,ip_address2|interface2@port2....] [-b port to bind] [-i] [-S <server name>] [-D <instance path>] [-d <path for dib>] [-m <module>] [-e] [-R|-r] [-c] [-L <ldap port>] [-l <SSL port>] [-P <LDAP URLs>] [-o http port] [-O https port] [--config-file <absolute path for configuration file>] [--configure-eba-now <yes/no>] ndsconfig add [-t <treename>] [-n <server context>] [-a <admin FDN>] [-w <password>] [-B ip_address1|interface1@port1,ip_address2|interface2@port2....] [-b port to bind] [-E] [-e] [-R|-r] [-c] [-L <ldap port>] [-l <SSL port>] [-P <LDAP URLs>] [-o http port] -O [https port] [-S <server name>] [-D <instance path >] [-d <path for dib>] [-p <IP address[:port]>] [-m <module>] [--config-file <absolute path for configuration file>] [--configure-eba-now <yes/no>] ndsconfig rm [-a <*admin FDN*>] [-w <*admin password*>] [-W <*obfuscated_password_file*>] [-c] [--config-file <*configuration file*>] ndsconfig upgrade [-a <admin FDN>] [-w <password>] [-c] [-j] [--config-file <absolute path for configuration file>] [--configure-eba-now <yes/no>] ndsconfig {set <*valuelist*> | get [<*paramlist*>] | get help [<*paramlist*>]}

Command	Description	Usage
ndscheck	Utility that checks the health of the tree.	ndscheck [--help | -?] Display command usage ndscheck [--version | -v] Display version information ndscheck [-h <*hostname port*>] [-a <*admin FDN*>] [-F <*log file*>] [-D] [-q] [-w <*admin password*>] [-W] [--config-file <*file name*>] ndscheck [-a <*admin FDN*>] [-W] [--config-file <*file name*>] For example: ndscheck -a admin.novell -W --config-file /etc/opt/novell/eDirectory/conf-1/nds.conf
ndsmanage	Utility that lists the OpenText eDirectory instances.	ndsmanage [-a] ndsmanage [<*username*>]

Command	Description	Usage
ndsbackup	Creates OpenText eDirectory object archives and adds or extracts OpenText eDirectory objects	ndsbackup c [f <*ndsbackupfile*>] [e] [v] [w] [X<*exclude-file*>] [R] [Replica-server-name] [-a <*admin-user*>] [-I <*include-file*>] [-E <*password*>] [--config-file <*configuration_file_path*>]... [eDirectoryobject] ndsbackup r [f <*ndsbackupfile*>] [e] [v] [w] [X<*exclude-file*>] [R] [Replica-server-name] [-a <admin-user>] [-I <*include-file*>] [-E <*password*>] [--config-file <*configuration_file_path*>]... [eDirectoryobject] ndsbackup t [f <*ndsbackupfile*>] [e] [v] [w] [X<*exclude-file*>] [R] [Replica-server-name] [-a <admin-user>] [-I <*include-file*>] [-E <*password*>] [--config-file <configuration_file_path>]... [eDirectoryobject] ndsbackup x [f <*ndsbackupfile*>] [e] [v] [w] [X<*exclude-file*>] [R] [Replica-server-name] [-a <admin-user>] [-I <*include-file*>] [-E <*password*>] [--config-file <*configuration_file_path*>]... [eDirectoryobject] ndsbackup s [e] [v] [w] [X<*exclude-file*>] [R] [Replica-server-name] [-a <*admin-user*>] [-I <*include-file*>] [-E <*password*>] [--config-file <*configuration_file_path*>]... [eDirectoryobject] ndsbackup --version ndsbackup [option] [file] [-a <*admin FDN*>] [-p passstore] [--config-file <*file name*>] For example: ndsbackup cvf /tmp/test.bak -a admin.novell -p passstore --config-file /etc/opt/novell/eDirectory/conf-1/nds.conf
ndslogin	Diagnostic utility to verify OpenText eDirectory authentication	ndslogin [-t treename] [-p password] [-s] [-n] [-c] [[-i] | [-I]] [[-h hostname[:port]] | [--config-file <configuration file>]] <userFDN>

Command	Description	Usage
ndsd	NDS daemon	/opt/novell/eDirectory/sbin/ndsd [--config-file *configfile*]
ndsimonitor	Monitors and diagnoses the servers in the OpenText eDirectory tree using HTTP	/opt/novell/eDirectory/bin/ndsimonitor [-l [-d <*path of ndsimonitor conf files*>] | u] [-h <*local_interface:port*>] [--config-file <*configuration_file_path*>]
ndsmerge	Utility to merge two OpenText eDirectory trees	ndsmerge [-m target-tree target-admin source-admin [target-container]] [-c] [-t] [-r target-tree source-admin] [-h <*local_interface:port*>] [--config-file <*configuration_file_path*>]
ndsrepair	Utility to repair and correct problems with the OpenText eDirectory database, such as records, schema, bindery objects, and external references.	ndsrepair {-U |-E |-C |-P [Ad] |-S [Ad]|-N |-T |-J <*entry_id*>} [-A <*yes/no*>] [-O <*yes/no*>][-F <*filename*>] [-h <*local_interface:port*>] [--config-file <*configuration_file_path*>]
		ndsrepair -R [-l <*yes/no*>][-u <*yes/no*>][-m <*yes/no*>][-i <*yes/no*>][-f <*yes/no*>][-d <*yes/no*>][-t <yes/no>][-o <yes/no>][-r <yes/no>][-v <yes/no>][-c <yes/no>][-A <yes/no>][-O <yes/no>][-F <*filename*>] [-h <*local_interface*>] [--config-file <*configuration_file_path*>]
	You can instruct ndsrepair to display information about the free space in the database that can be released for your use.	ndsrepair –I [--config-file <*configuration_file_path*>]
ndssch	OpenText eDirectory schema extension utility	ndssch [-h <*hostname*>[:<port>]][-t <*treename*>][-F <*logfile*>] <*admin-FDN*> <*schemafile*> ...
		ndssch [-h <*hostname*>[:<port>]][-t <*treename*>] [-d] <*admin-FDN*> <*schemafile*> [schema description] ...
ndssnmp	SNMP services module for OpenText eDirectory.	/opt/novell/eDirectory/bin/ndssnmp
ndssnmpconfig	SNMP trap configuration utility	ndssnmpconfig [-h <*hostname[:port]*>] [-p <*password*>] [-a <*userFDN*>] [-c <*command*>]
ndssnmpsa	OpenText eDirectory SNMP subagent daemon	/opt/novell/eDirectory/bin/ndssnmpsa

Command	Description	Usage
ndsstat	Utility that displays the server information	ndsstat { -r -s -p <*partitionname*>} [-n] [[-h <*hostname* | *IP address*>:<*port*>] | [--config-file <*configuration file*>]]
ndstrace	Utility that displays the server debug messages	ndstrace [-l|-u|-c "command1;....."|--version] [-h <*local_interface:port*>] [--config-file <*configuration_file_path*>]
nds-uninstall	Utility to uninstall OpenText eDirectory	nds-uninstall [-s][-h]
nldap	LDAP services for NDS daemon	/opt/novell/eDirectory/sbin/nldap
nmasinst	NMAS configuration utility	nmasinst -i <*admin-FDN*> <*treename*> [-h <*hostname*>[:port]] nmasinst -addmethod <*admin-FDN*> <*treename*> <*config.txt file*> [-h <*hostname*>[:port]]
npki	Novell Public Key Infrastructure Services	/opt/novell/eDirectory/sbin/npki

LDAP-Specific Commands

Command	Description	Usage
ldapconfig	Utility to configure LDAP Server and LDAP Group objects	ldapconfig get [...] | set <*attribute-value-list*> [-t <*treename*> | -p <*hostname*>[:port] | --config-file <*configuration file*>] [-w <*password*>] [-a <*user FDN*>] [-f] ldapconfig [-t <*treename*> | -p <*hostname*>[:port]] [-w <*password*> | --config-file <*configuration file*>] [-a <*user FDN*>] [-V] [-R] [-H] [-f] -v <*attribute*>,<*attribute2*>... ldapconfig [-t <*treename*> | -p hostname[:port] | --config-file <*configuration file*>] [-w <*password*>] [-a <*admin FDN*>] [-V] [-R] [-H] [-f] -s <*attribute*>=<*value*>,...

Command	Description	Usage
ldapadd ldapmodify	Add or modify entries from an LDAP server	ldapmodify [-a] [-c] [-C] [-M] [-P] [-r] [-n] [-v] [-F] [-l <limit>] [-M[M]] [-d <debuglevel>] [-e <key filename>] [-D <binddn>] [[-W]|[-w <passwd>]] [-H ldap://[ldap host ip]:[ldap-port]] [-P <version>] [-Z[Z]] [-f <file>] ldapadd [-c] [-C] [-l] [-M] [-P] [-r] [-n] [-v] [-F] [-l <*limit*>] [-M[M]] [-d <*debuglevel*>] [-e <*key filename*>] [-D <*binddn*>] [[-W]| [-w <*passwd*>]] [-H ldap://[ldap host ip]:[ldap-port] [-P <*version*>] [-Z[Z]] [-f <*file*>]
ldapdelete	Delete entries from an LDAP server	ldapdelete [-n] [-v] [-c] [-r] [-l] [-C] [-M] [-d <*debuglevel*>] [-e <*key filename*>] [-f <*file*>] [-D <*binddn*>] [[-W]| [-w <*passwd*>]] [-H ldap://[ldap host ip]:[ldap-port]} [-Z[Z]] [dn]...
ldapmodrdn	LDAP modify entry Relative Distinguished Name (RDN) tool.	ldapmodrdn [-r] [-n] [-v] [-c] [-C] [-l] [-M] [-s <*newsuperior*>] [-d <*debuglevel*>] [-e <*key filename*>] [-D <*binddn*>] [[-W]|[-w <*passwd*>]] [-h <*ldaphost*>] [-p <*ldapport*>] [-Z[Z]] [-f <*file*>] [dn <*newrdn*>]
ldapsearch	The LDAP search tool	ldapsearch [-n] [-u] [-v] [-t] [-A] [-T] [-C] [-V] [-M] [-P] [-L] [-d <*debuglevel*>] [-e <*key filename*>] [-f <*file*>] [-D <*binddn*>] [[-W]| [-w <*bindpasswd*>]] [-h <*ldaphost*>] [-p <*ldapport*>] [-b <*searchbase*>] [-s <*scope*>] [-a <*deref*>] [-l <*time limit*>] [-z <*size limit*>] [-Z[Z]] filter [attrs....]

Command	Description	Usage
ndsindex	Utility to create, list, suspend, resume, or delete OpenText eDirectory database indexes.	ndsindex list [-h <*hostname*>] [-p <*port*>] [-D <*bind DN*>*] [-W|[-w <*password*>]] [-l <*limit*>] [-s <*eDirectory Server DN*>] [-Z[Z]] [<*indexName1*>, <*indexName2*>.....] ndsindex add [-h <*hostname*>] [-p <*port*>] [-D <*bind DN*>] -W|[-w <*password*>] [-l <*limit*>] [-s <*eDirectory Server DN*>] [-Z[Z]] <*indexDefinintion1*> [<*indexDefinintion2*>.....] ndsindex delete [-h <*hostname*>] [-p <*port*>] [-D <*bind DN*>] [-W|[-w <*password*>]] [-l <*limit*>] [-s <*eDirectory Server DN*>] [-Z[Z]] <*indexName1*> [<*indexName2*>.....] ndsindex resume [-h <*hostname*>] [-p <*port*>] -D <*bind DN*> [-W|[-w <*password*>]] [-l <*limit*>] [-s <*eDirectory Server DN*>] [-Z[Z]] <*indexName1*> [<*indexName2*>.....] ndsindex suspend [-h <*hostname*>] [-p <*port*>] [-D <*bind DN*>] [-W|[-w <*password*>]] [-l <limit>] [-s <*eDirectory Server DN*>] [-Z[Z]] <*indexName1*> [<*indexName2*>.....]
ice	Utility to Import entries from a file to an LDAP directory, to modify the entries in a directory from a file, to export the entries to a file, and to add attribute and class definitions from a file.	ice -S LDAP -s server1.acme.com -p 636 -L cert-server1.pem -d cn=admin,c=us -w password -F objectClass=* -c sub -D LDIF -f server1.ldif -e des -E secret ice -S LDIF -f server1.ldif -e des -E secret -D LDAP -s server2.acme.com -p 636 -L cert-server2.pem -d cn=admin,c=us -w password

Special Characters in User Name and Password

Using special characters in user names and passwords can create problems when the values are passed during an OpenText eDirectory installation or schema extension. If the user name or password contains special characters, such as $, # and so on, escape the character by preceding it with a backslash (\) .

For example, an administrator user name of `cn=admin$name.o=container` must be passed as `cn=admin\$name.o=container`.

When entering parameter values at the command line, you can escape the character, or place single quotes around the value.

For example,

```
cn=admin\$name.o=container
```

or

```
'cn=admin$name.o=container'
```

C Configuring OpenSLP for OpenText eDirectory

This appendix provides information for network administrators on the proper configuration of OpenSLP for OpenText eDirectory installations without the Novell Client.

- "Service Location Protocol" on page 775
- "SLP Fundamentals" on page 775
- "Configuration Parameters" on page 777

Service Location Protocol

OpenSLP is an open-source implementation of the IETF Service Location Protocol Version 2.0 standard, which is documented in IETF Request-For-Comments (RFC) 2608 (http://www.ietf.org/rfc/rfc2608.txt?number=2608).

In addition to implementing the SLP v2 protocol, the interface provided by OpenSLP source code is an implementation of another IETF standard for programmatically accessing SLP functionality, documented in RFC 2614 (http://www.ietf.org/rfc/rfc2614.txt?number=2614).

To fully understand the workings of SLP, we recommend that you read these two documents and internalize them. They are not necessarily light reading, but they are essential to the proper configuration of SLP on an intranet.

For more information on the OpenSLP project, see the OpenSLP (http://www.OpenSLP.org) Web site and the SourceForge (http://sourceforge.net/projects/openslp) Web site. The OpenSLP Web site provides several documents that contain valuable configuration tips. Many of these are incomplete at the time of this writing.

SLP Fundamentals

Service Location Protocol specifies three components:

- The user agent (UA)
- The service agent (SA)
- The directory agent (DA)

The user agent's job is to provide a programmatic interface for clients to query for services, and for services to advertise themselves. A user agent contacts a directory agent to query for registered services of a specified service class and within a specified scope.

The service agent's job is to provide persistent storage and maintenance points for local services that have registered themselves with SLP. The service agent essentially maintains an in-memory database of registered local services. In fact, a service cannot register with SLP unless a local SA is present.

Clients can discover services with only a UA library, but registration requires an SA, primarily because an SA must reassert the existence of registered services periodically in order to maintain the registration with listening directory agents.

The directory agent's job is to provide a long-term persistent cache for advertised services, and to provide a point of access for user agents to look up services. As a cache, the DA listens for SAs to advertise new services, and caches those notifications. Over a short time, a DA's cache will become more complete. Directory agents use an expiration algorithm to expire cache entries. When a directory agent comes up, it reads its cache from persistent storage (generally a hard drive), and then begins to expire entries according to the algorithm. When a new DA comes up, or when a cache has been deleted, the DA detects this condition and sends out a special notification to all listening SAs to dump their local databases so the DA can quickly build its cache.

In the absence of any directory agents, the UA will resort to a general multicast query that SAs can respond to, building a list of the requested services in much the same manner that DAs use to build their cache. The list of services returned by such a query is an incomplete and much more localized list than that provided by a DA, especially in the presence of multicast filtering, which is done by many network administrators, limiting broadcasts and multicasts to only the local subnet.

In summary, everything hinges on the directory agent that a user agent finds for a given scope.

OpenText Service Location Providers

The OpenText version of SLP takes certain liberties with the SLP standard in order to provide a more robust service advertising environment, but it does so at the expense of some scalability.

For example, in order to improve scalability for a service advertising framework, we want to limit the number of packets that are broadcast or multicast on a subnet. The SLP specification manages this by imposing restrictions on service agents and user agents regarding directory agent queries. The first directory agent discovered that services the desired scope is the one that a service agent (and consequently, local user agents) will use for all future requests on that scope.

The OpenText SLP implementation actually scans all of the directory agents it knows about looking for query information. It assumes a 300-millisecond round trip time is too long, so it can scan 10 servers in about 3 to 5 seconds. This doesn't need to be done if SLP is configured correctly on the network, and OpenSLP assumes the network is in fact configured correctly for SLP traffic. OpenSLP's response timeout values are greater than that of OpenText's SLP service provider, and it limits the number of directory agents to the first one that responds, whether or not that agent's information is accurate and complete.

User Agents

A user agent takes the physical form of a static or dynamic library that is linked into an application. It allows the application to query for SLP services.

User agents follow an algorithm to obtain the address of a directory agent to which queries will be sent. Once they obtain a DA address for a specified scope, they continue to use that address for that scope until it no longer responds, at which time they obtain another DA address for that scope. User agents locate a directory agent address for a specified scope by:

1. Checking to see if the socket handle on the current request is connected to a DA for the specified scope. If the request happens to be a multipart request, there may already be a cached connection present on the request.

2. Checking its local known DA cache for a DA matching the specified scope.
3. Checking with the local SA for a DA with the specified scope and adding new addresses to the cache.
4. Querying DHCP for network-configured DA addresses that match the specified scope and adding new addresses to the cache.
5. Multicasting a DA discovery request on a well-known port and adding new addresses to the cache.

The specified scope is “default” if not specified. That is, if no scope is statically defined in the SLP configuration file, and no scope is specified in the query, then the scope used is the word “default”. It should also be noted that OpenText eDirectory never specifies a scope in its registrations. That’s not to say the scope always used with OpenText eDirectory is “default.” In fact, if there is a statically configured scope, that scope becomes the default scope for all local UA requests and SA registrations in the absence of a specified scope.

Service Agents

Service agents take the physical form of a separate process on the host machine. In the case of Windows, `slpd.exe` runs as a service on the local machine. User agents query the local service agent by sending messages to the loop-back address on a well-known port.

A service agent locates and caches directory agents and their supported scope list by sending a DA discovery request directly to potential DA addresses by:

1. Checking all statically configured DA addresses (and adding new ones to the SA’s known DA cache).
2. Requesting a list of DA’s and scopes from DHCP (and adding new ones to the SA’s known DA cache).
3. Multicasting a DA discovery request on a well-known port (and adding new ones to the SA’s known DA cache).
4. Receiving DA advertising packets that are periodically broadcast by DAs (and adding new ones to the SA’s known DA cache).

Since a user agent always queries the local service agent first, this is important, as the local service agent’s response will determine whether or not the user agent continues to the next stage of discovery (in this case DHCP-- see steps 3 and 4 in “User Agents” on page 776.).

Configuration Parameters

The SLP configuration parameters are stored in the `slp.conf` file, located in `/etc` on UNIX and Linux platforms and `%systemroot%/slp.conf` on Windows platforms. These parameters can be modified to tune the network operations.For example, the following parameters control the DA discovery:

```
net.slp.useScopes = <comma-delimited scope list>
net.slp.DAAddresses = <comma-delimited address list>
net.slp.passiveDADetection = <"true" or "false">
net.slp.activeDADetection = <"true" or "false">
net.slp.DAActiveDiscoveryInterval = <0, 1, or a number of seconds>
```

The useScopes option indicates which scopes the SA will advertise into, and which scopes queries will be made to in the absence of a specific scope on the registration or query made by the service or client application. Because OpenText eDirectory always advertises into and queries from the default scope, this list will become the default scope list for all OpenText eDirectory registrations and queries.

The DAAddresses option is a comma-delimited list of dotted decimal IP addresses of DAs that should be preferred to all others. If this list of configured DAs does not support the scope of a registration or query, then SAs and UAs will resort to multicast DA discovery, unless such discovery is disabled.

The passiveDADetection option is True by default. Directory agents will periodically broadcast their existence on the subnet on a well-known port if configured to do so. These packets are termed DAAdvert packets. If this option is set to False, all broadcast DAAdvert packets are ignored by the SA.

The activeDADetection option is also True by default. This allows the SA to periodically broadcast a request for all DAs to respond with a directed DAAdvert packet. A directed packet is not broadcast, but sent directly to the SA in response to these requests. If this option is set to False, no periodic DA discovery request is broadcast by the SA.

The DAActiveDiscoveryInterval option is a try-state parameter. The default value is 1, which is a special value meaning that the SA should only send out one DA discovery request upon initialization. Setting this option to 0 has the same effect as setting the activeDADetection option to “false.” Any other value is a number of seconds between discovery broadcasts.

These options, when used properly, can ensure an appropriate use of network bandwidth for service advertising. In fact, the default settings are designed to optimize scalability on an average network.

NOTE: By default, the IPV4 protocol is enabled for SLP and IPV6 is disabled. To enable IPV6, uncomment the following line in the `slp.conf` file:

```
net.slp.useIPv6 = true
```

This is valid only for Windows because OpenSLP 2.0 is shipped only for Windows.

---

slptool Utility

This is a command line utility provided by OpenSLP. You can use slptool to register or de-register the services, query the scopes, service types, attributes, and the services available.

For example:

- To register the services,

  Syntax: `slptool register url` *`[attrs]`*

  ```
  slptool register service:myserv.x://myhost.com
  "(attr1=val1),(attr2=val2)"
  ```

- To de-register a service,

  Syntax: `slptool deregister` *`url`*

  ```
  slptool deregister service:myserv.x://myhost.com
  ```

- To find the available services,

  Syntax: `slptool findsrvs` *`service-type [filter]`*

```
slptool findsrvs service:myserv.x
slptool findsrvs service:myserv.x "(attr1=val1)"
```

- To find the configured scopes,
  Syntax: `slptool findscopes`

D How OpenText eDirectory Works with DNS

If a client asks a server to resolve a fully qualified name (for example, admin.novell.novell_inc) that does not exist in the OpenText eDirectory tree, or if you use a standalone application such as OpenText Identity Console for Linux or the OpenText eDirectory install application to resolve a name in the tree and you don't have a server to talk to yet, OpenText eDirectory uses service discovery protocols to resolve the name. Service discovery protocols are a class of network applications that allow distributed components to find and use needed services within a network.

OpenText eDirectory has traditionally used SAP and SLP to search for and advertise network services. DNS was added as a discovery protocol in OpenText eDirectory 8.7.1. This added functionality means that if you ask for a tree name that OpenText eDirectory doesn't understand (either because you are talking to a server that doesn't hold a copy of the tree or you are using a stand-alone application), the machine trying to do the discovery—whether it's a machine running a stand-alone application, a JClient application such as OpenText Identity Console or a server—uses OpenText eDirectory's discovery protocols, in the following order:

1. Domain Name System (DNS)
2. Service Location Protocol (SLP)
3. Service Advertising Protocol (SAP)

When using the DNS protocol, OpenText eDirectory takes the name as it was passed (for example, a server name such as prod_server4.provo.novell.novell-_inc), and tries to resolve the entire name just as it is. OpenText eDirectory then appends each name in the discovery machine's DNS search list and asks the machine's DNS server if it has an address for that name. For example, if the discovery machine's DNS search list included dev.novell.com and test.novell.com, OpenText eDirectory would search for prod_server4.provo.novell.novell_inc.dev.novell.com and prod_server4.provo.novell.novell_inc.test.novell.com.

Then OpenText eDirectory takes components off the name that was passed to it. For example, if trying to resolve prod_server4.provo.novell.novell_inc, OpenText eDirectory tries provo.novell.novell_inc, then novell.novell_inc, then novell_inc. OpenText eDirectory does that for each of the different search contexts until eventually it tries the single component that is the tree root. The client will attempt each of the addresses until it successfully makes a connection. It does the attempts using the ordering of records returned from the DNS server. It doesn’t matter what code revision the servers in the replica ring are running as long as the machine trying to do the discovery is running OpenText eDirectory 8.7.1 or later.

We recommend putting your OpenText eDirectory tree name in DNS using an A, AAAA, or Service (SRV) resource record under the DNS domain the clients are going to use to resolve names. If you use A or AAAA records, the OpenText eDirectory servers must be running on the default 524 port. If the servers are using any other port, use an SRV record.

In the following sample resource records, novell_inc is the tree name and provo.novell.com is the DNS search context:

Record	Example
A	novell_inc.provo.novell.com. IN A 192.168.1.2
AAAA	novell_inc.provo.novell.com. IN AAAA 4321:0:1:2:3:4:567:89ab
SRV	_ldap._tcp.novell_inc.provo.novell.com. SRV 0 0 389 server1.novell_inc.provo.novell.com SRV 10 0 389 server2.novell_inc.provo.novell.com

For redundancy, or to specify multiple hosts (servers in the replica ring) to the A record, create more than one A record. OpenText eDirectory will look at all of them. For more information on A, AAAA, and SRV records, see “DNS resource records”.

You don't need to point the DNS server record entry to something that holds a corresponding partition root. As soon as the discovery machine can talk to a server that knows about the tree, it can walk up and down the tree to resolve the name. For example, if you put novell_inc in your DNS, you don't have to put in any of the servers that hold novell_inc root. All you need to do is point to any server in the novell_inc tree, because after you get to that server in the tree, that server will refer you around the tree.

E Configuring GSSAPI with OpenText eDirectory

The SASL-GSSAPI mechanism for OpenText eDirectory enables you to authenticate to OpenText eDirectory through LDAP using a Kerberos ticket. You are not required to enter the OpenText eDirectory user password. The Kerberos ticket must be obtained by authenticating to a Kerberos server.

This feature is primarily useful for LDAP application users in environments that already have a Kerberos infrastructure in place. Therefore, these users should be able to authenticate to the LDAP server without providing a separate LDAP user password.

The current implementation of SASL-GSSAPI is compliant with RFC 2222 (http://www.ietf.org/rfc/rfc2222.txt?number=2222) and supports only Kerberos v5 as the authentication mechanism.

The following sections explain how to configure GSSAPI and describe the various tasks you can perform with Kerberos in OpenText eDirectory and give some useful additional information:

- "Concepts" on page 783
- "How Does GSSAPI Work with OpenText eDirectory?" on page 784
- "Prerequisites for Configuring GSSAPI" on page 785
- "Merging OpenText eDirectory Trees Configured with SASL-GSSAPI Method" on page 787
- "Managing the SASL-GSSAPI Method" on page 788
- "Creating a Login Sequence" on page 789
- "How Does LDAP Use SASL-GSSAPI?" on page 789
- "Error Messages" on page 789
- "Commonly Used Terms" on page 789

Concepts

- "What is Kerberos?" on page 783
- "What is SASL?" on page 784
- "What is GSSAPI?" on page 784

What is Kerberos?

Kerberos is a standard protocol that provides a means of authenticating entities on a network. It is based on a trusted third-party model. It involves shared secrets and uses symmetric key cryptography.

For more information, refer to RFC 1510 (http://www.ietf.org/rfc/rfc1510.txt?number=1510).

What is SASL?

Simple Authentication and Security Layer (SASL) provides an authentication abstraction layer to applications. It is a framework that authentication modules can be plugged into.

For more information, refer to RFC 2222 (http://www.ietf.org/rfc/rfc2222.txt?number=2222).

What is GSSAPI?

Generic Security Services Application Program Interface (GSSAPI) provides authentication and other security services through a standard set of APIs. It supports different authentication mechanisms. Kerberos v5 is the most common.

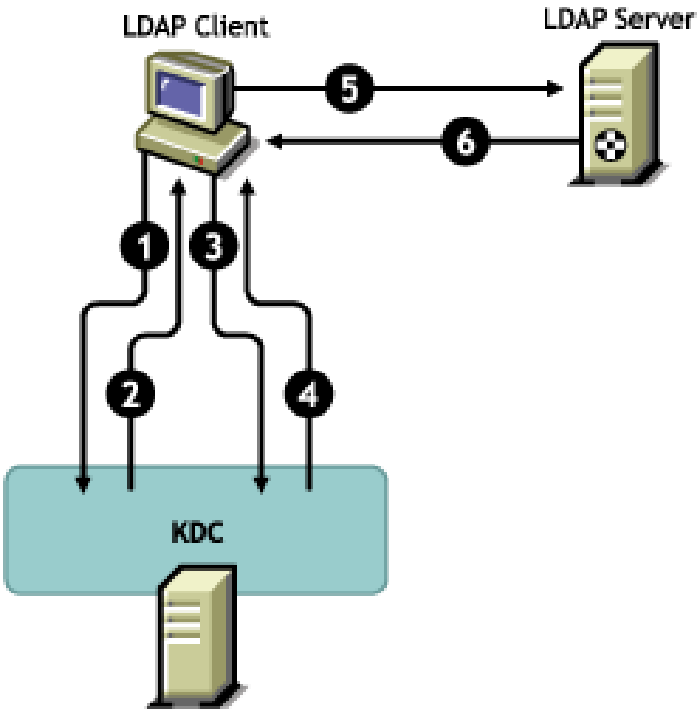
For more information on the GSS APIs, refer to RFC 1964 (http://www.ietf.org/rfc/rfc1964.txt?number=1964).

This SASL-GSSAPI implementation is from section 7.2 of RFC 2222 (http://www.ietf.org/rfc/rfc2222.txt?number=2222).

How Does GSSAPI Work with OpenText eDirectory?

The following diagram illustrates how GSSAPI works with an LDAP server.

Figure E-1 *How GSSAPI Works?*



In the above figure, the numbers denote the following:

1. An OpenText eDirectory user sends a request through an LDAP client to the Kerberos KDC (Key Distribution Center) server for an initial ticket known as a ticket granting ticket (TGT).

A Kerberos KDC can be from MIT or Microsoft*.

2. KDC responds to the LDAP client with a TGT.
3. The LDAP client sends the TGT back to the KDC and requests an LDAP service ticket.
4. KDC responds to the LDAP client with the LDAP service ticket.
5. The LDAP client does an ldap_sasl_bind to the LDAP server and sends the LDAP service ticket.
6. The LDAP server validates the LDAP service ticket with the help of the GSSAPI mechanism and, based on the result, sends back an ldap_sasl_bind success or failed to the LDAP client.

Prerequisites for Configuring GSSAPI

To configure GSSAPI, you must first do the following:

- **SASL-GSSAPI method:** Install the SASL-GSSAPI method. Refer to the Installing a Login Method section in the Modular Authentication Services (NMAS Functionality).

  NOTE: The OpenText eDirectory SASL-GSSAPI method does not work on installations of Open Enterprise Server versions 2 or 11 that have Domain Services for Windows installed.

  To verify whether SASL-GSSAPI is installed on your machine, enter the following:

  ```
  ldapsearch -x -h osg-dt-srv9 -b " " -s base | grep -i sasl
  ```

  If SASL-GSSAPI is installed, the output of the command is similar to the following:

  ```
  supportedSASLMechanisms: NMAS_LOGIN

  supportedSASLMechanisms: GSSAPI
  ```

- **Key distribution center (KDC):** Install Kerberos KDC (MIT; Active Directory) on the network.

  For Microsoft KDC (Active Directory), you must have the Kerberos tools installed. These tools are part of the Windows installation and can be installed from `\support\tools\setup.exe` (Windows XP) and `\support\tools\suptools.msi` (Windows 2003) on the Windows installation CD.

- **Time Synchronization:** Synchronize the time on the NMAS client machine, the NMAS server machine, and the KDC machine for this method to work. For more information on synchronizing network time, refer to "Synchronizing Network Time" on page 92.

- **Kerberos LDAP Extensions:** Add the Kerberos LDAP extensions. For more information, see "Adding Kerberos LDAP Extensions" on page 786.

  IMPORTANT:
  - On Open Enterprise Server, do not add the Kerberos LDAP extensions on servers where Domain Services for Windows or DNS services are configured.
  - All Kerberos information collected from your Kerberos administration is case-sensitive and must be specified exactly in the same case.

Assumptions on Network Characteristics

The SASL-GSSAPI mechanism is based on the following assumptions:

- All the machines in the network have loosely synchronized time. This means that no two machines in the network have their system time differing by more than five minutes.
- The SASL-GSSAPI mechanism is expected to be used mostly in LAN as it is difficult to obtain the time synchronization requirement mentioned above in MAN and/or WAN environments. However, this mechanism is not limited to LAN.
- You trust the Kerberos servers and Kerberos administrators unconditionally and unverifiably.
- Denial-of-Service attack is not countered. For more information, refer to RFC 1510 (http://www.ietf.org/rfc/rfc1510.txt?number=1510).

Adding Kerberos LDAP Extensions

Kerberos LDAP Extensions provide the functionality to manage Kerberos keys.

To use the Kerberos LDAP extensions, you must install the LDAP libraries for C. For more information, refer to LDAP Libraries for C (https://www.microfocus.com/documentation/edirectory-developer-documentation/ldap-libraries-for-c/).

To add or remove the Kerberos LDAP extensions, use the krbLdapConfig utility. When standalone OpenText eDirectory package is extracted to a directory, the path of this file is `extracted_folder/nmas/NmasMethods/Novell/GSSAPI/Kerberos_ldap_extensions/Linux/krbLdapConfig`.

For example, `/misc/eDir88/Linux/nmas/NmasMethods/Novell/GSSAPI/Kerberos_ldap_extensions/Linux/krbLdapConfig`.

To add the Kerberos LDAP extensions, use the following syntax:

```
krbldapconfig {-i | -u} -D bind_DN [-w bind_DN_password] [-H ldap://[ldap host ip]:[ldap-port] [-e trusted_root_cert]
```

The following table explains the krbldapconfig utility parameters:

Parameter	Description
`-i`	Adds the Kerberos LDAP extensions to OpenText eDirectory.
`-u`	Removes the Kerberos LDAP extensions from OpenText eDirectory.
`-D bind_fdn`	Specifies the FDN of the administrator or the user with administrator-equivalent rights. This must be in the format `cn=admin,o=org`.
`-w bind_fdn_password`	Specifies the password of the bind FDN (`bind_fdn`).
`-h ldap_server`	Specifies the hostname or IP address of the LDAP server where Kerberos LDAP extensions must be installed.
`-p port`	Specifies the port where the LDAP server is running.

Parameter	Description
`-e trusted_root_file`	Specifies the trusted root certificate filename for the SSL bind. If you are using an SSL port, specify the `-e` option. For more information, refer to "Exporting the Trusted Root Certificate" on page 787.

NOTE: If you do not specify the `-h` option, the name of the local host that krbldapconfig is invoked from is used as the default.

If you do not specify the LDAP server port and the trusted root certificate, the default port 389 is used.

If you do not specify the LDAP server port but specify the trusted root certificate, the default port 636 is used.

For example, enter the following to add the extensions:

```
krbldapconfig -i -D cn=admin,o=org -w password -h ldapserver -p 389
```

Or to remove, enter the following:

```
krbldapconfig -u -D cn=admin,o=org -w password -h ldapserver -p 389
```

IMPORTANT: You must manually refresh the LDAP server for the installation changes to take effect. For more information, refer to "Refreshing the LDAP Server" on page 383.

Exporting the Trusted Root Certificate

1. On the OpenText Identity Console home page, click **Certificate management** tile > **Trusted Root Management** to open the Modify Object page.
2. Click **Trusted Roots** tab, then select **Trusted Root Certificate** and view the details of the certificate.
3. Click **Export** .
4. Specify whether you want to export the private key or not. If you want to export the private key, you might need to specify a password to protect the private key.
5. Click **OK**.
6. Click **Save the exported certificate.**
7. Click **Save File**.
8. Click **Close**.

Merging OpenText eDirectory Trees Configured with SASL-GSSAPI Method

When you merge two trees, either or both configured with the SASL-GSSAPI method, you need to manually create all the Kerberos objects that are in the source tree in the target tree.

Managing the SASL-GSSAPI Method

You can perform the following Kerberos operations:

- “Managing a Service Principal” on page 788

Managing a Service Principal

This section discusses the following:

- “Creating a Service Principal for an LDAP Server” on page 788
- “Extracting the Key of the Service Principal for OpenText eDirectory” on page 788

Creating a Service Principal for an LDAP Server

Use the Kerberos Administration tool that is available with your KDC to create the OpenText eDirectory service principal with the encryption type and salt type as AES256-CTS and Normal, respectively.

The name of the principal must be ldap/*MYHOST.MYDNSDOMAIN@REALMNAME*.

For example, if you are using MIT KDC, execute the following command:

```
kadmin:addprinc -randkey -e aes256-cts:normal ldap/
server.novell.com@MITREALM
```

IMPORTANT: The hostname of service principal created must be in lowercase. Authentication fails if the hostname is in uppercase. For example, if the hostname is myHost.com, the hostname syntax of the LDAP service principal should look like `ldap/myhost.com@<realmname>`.

Best Practice

- All the keys should be preferably of type AES256.
- Change the LDAP service principal keys regularly. Whenever you change the LDAP service principal keys, ensure that you update the principal object in OpenText eDirectory.

Extracting the Key of the Service Principal for OpenText eDirectory

Use the Kerberos Administration tool that is available with your KDC to extract the key of the LDAP service principal created in “Creating a Service Principal for an LDAP Server” on page 788, then store it in the local file system. This can be done with the help of your Kerberos administrator.

For example, if you are using an MIT KDC, execute the following command:

```
kadmin: ktadd -k /directory_path/keytabfilename -e aes256-cts:normal ldap/
server.novell.com@MITREALM
```

For example, if you are using Microsoft KDC, create a user ldapMYHOST in Active Directory and then execute the following command:

```
ktpass -princ ldap/MYHOST.MYDNSDOMAIN@MYREALM -mapuser ldapMYHOST -pass
mypassword -out MYHOST.keytab
```

This command maps the principal (ldap/MYHOST.MYDNSDOMAIN@MYREALM) to the user account (ldapMYHOST), sets the host principal password to `mypassword`, and extracts the key into the `MYHOST.keytab` file.

Creating a Login Sequence

For information on creating a login sequence, refer to the "Managing Login and Post-Login Methods and Sequences" on page 609.

How Does LDAP Use SASL-GSSAPI?

Once you have configured SASL-GSSAPI, it is added along with the other SASL methods to the supportedSASLMechanisms attribute in rootDSE.

The LDAP server queries SASL for the installed mechanisms when it gets its configuration, and automatically supports whatever is installed. The LDAP server also reports the current supported SASL mechanisms in its rootDSE by using the supportedSASLMechanisms attribute.

Therefore, once you configure GSSAPI, it becomes the default mechanism.

However, to specifically do an LDAP operation over the SASL GSSAPI mechanism, you can mention GSSAPI at the command line.

For example, in OpenLDAP to do a search using the GSSAPI mechanism, enter the following:

```
ldapsearch -H ldap://10.10.10.10:389  -b "" -s base "objectclass=*""
```

Error Messages

The SASL-GSSAPI error messages are logged into the following locations:

- Linux: `ndsd.log`

For more information, "Managing Error Logging in OpenText eDirectory" on page 845.

Commonly Used Terms

The following table defines the terminologies commonly used with Kerberos and GSSAPI.

Table E-1 *Kerberos/GSSAPI Terminology*

Term	Definition
Key Distribution Center (KDC)	Kerberos server which authenticates users and issues tickets.
Principal	An entity (user or service instance) registered with the KDC.
Realm	A domain or grouping of principals served by a set of KDCs.

Term	Definition
Service Ticket (ST)	A record containing client information, service information, and a session key which is encrypted with the particular service principal's shared key
Ticket Granting Ticket (TGT)	A type of ticket that the client can obtain additional Kerberos tickets with.

F Security Considerations

This appendix contains the following topics:

- "LDAP Binds" on page 791
- "Nessus Scan Results" on page 791

LDAP Binds

The LDAP binds should take place over a secure connection. We recommend that you always use a SSL/TLS connection and keep in mind the following considerations:

- The key transmitted over the wire can be sniffed out. So you need to physically secure the corporate network against eaves-dropping or "packet sniffing".
- You need to keep the servers in a physically secure location with access by authorized personnel only.
- When the product is used by users outside of the corporate firewall, a VPN should be employed.
- If a server is accessible from outside the corporate network, a firewall should be configured to prevent direct access to the server.
- Audit logs should be checked periodically.
- Different administrative duties should be given to separate people. Delegation of administration provides granular control over the directory objects.
- We recommend that you identify a particular LDAP server as the right server for Kerberos management.

IMPORTANT: The user needs to access the LDAP server using the DNS name instead of the IP address of the server. This is because the conversion of the IP address to the DNS name is not secure.

Nessus Scan Results

The following vulnerabilities were reported by Nessus port scan:

- **LDAP servers that are not properly configured allow users to connect to the server and query for information**

  Explanation: Null Bind is enabled on OpenText eDirectory LDAP server by default but can be disabled on the server. To enhance the security of the server, disable the NULL bind on the LDAP server port 389. For more information, see "Configuring LDAP Objects" on page 369.

  Solution: Disable Null Bind on the server.

- **LDAP servers that are not properly configured set the directory base as null**

  Explanation: Information can be picked even without prior knowledge of the directory structure. With the help of Null Bind, an anonymous user can query the LDAP server using tools like “LdapMiner.”

  Solution: Although there is no way to disable it, security threat like this can be minimized by disabling Null Bind.

- **The remote service supports the use of weak SSL ciphers suites**

  Explanation: The remote host supports the use of SSL ciphers that offer either weak encryption or no encryption at all.

  Solution: Reconfigure the affected application, if possible, to avoid use of weak ciphers.

- **The remote directory server leaks information**

  Explanation: This host is a OpenText eDirectory server, and has Browse rights on the PUBLIC object.

  Solution: If applications using OpenText eDirectory do not depend on having PUBLIC rights, then assign the rights given to PUBLIC to authenticated users (ROOT) only. If this is an external system, it is recommended to block the access to port 524 from the Internet. However, tree and server name will be still be accessible even if you remove the Public browse rights.

- **SSL certificate is signed with an unknown certificate authority**

  Explanation: The X.509 certificate of the remote host is not signed by a known public certificate authority. If the remote host is a public host in production, this nullifies the use of SSL as anyone can establish a connection in the middle and attack against the remote host.

  Solution: This occurs when the client application does not have the certificate of the certificate authority that signed the server's certificate in its trusted certificate store. Purchase a certificate from a known certificate authority for the server and deploy it. Or, if the server's certificate has been issued either by the tree's organizational certificate authority or by an external or third-party certificate authority, then import or add the certificate authority's certificate in the applications trusted certificate store.

  For more information, see “Deciding Which Type of Certificate Authority to Use” on page 641.

G Mapping OpenText eDirectory Events with CEF Events

This section covers the following information:

- "Mapping OpenText eDirectory Events with CEF Events" on page 793
- "CEF Events" on page 797

Mapping OpenText eDirectory Events with CEF Events

Table G-1 lists OpenText eDirectory internal events mapped with the corresponding CEF events. For information about OpenText eDirectory events and their description, see OpenText eDirectory Services page. For information about CEF events, see "CEF Events" on page 797.

Table G-1 *CEF Events Mapped With OpenText eDirectory Events*

Event Category	CEF Event	OpenText eDirectory Event
Security	CONNECTION For an example of this event, see "Connection" on page 798.	DSE_CONNECTION
Security	LOGIN For an example of this event, see "Login" on page 798.	DSE_LOGIN_EX DSE_NMAS_LOG_SRVR_BEGIN_LOGIN DSE_NMAS_LOG_FINISH_LOGIN_STATUS DSE_NMAS_LOG_SASL_MECHANISM_RESULT DSE_EBA_REQ_BA_MATERIAL
Security	LOGOUT For an example of this event, see "Logout" on page 799.	DSE_LOGOUT
Security	ADD_MEMBER For an example of this event, see "Add Member" on page 799.	DSE_ADD_VALUE
Security	DELETE_MEMBER For an example of this event, see "Delete Member" on page 799.	DSE_DELETE_VALUE

Event Category	CEF Event	OpenText eDirectory Event
Security	INTRUDER_DETECTED	DSE_ADD_VALUE
	For an example of this event, see "Intruder Detected" on page 800.	
Security	ACCOUNT_UNLOCK	DSE_DELETE_VALUE
	For an example of this event, see "Account Unlock" on page 800.	
Security	LOGIN_DISABLED	DSE_ADD_VALUE
	For an example of this event, see "Login Disabled" on page 800.	
Security	LOGIN_ENABLED	DSE_DELETE_VALUE
	For an example of this event, see "Login Enabled" on page 801.	
Security	ACL_CHANGED	DSE_ADD_VALUE
	For an example of this event, see "ACL Changed" on page 801.	DSE_DELETE_VALUE
Security	CHANGE_SECURITY_EQUALS	DSE_ADD_VALUE
	For an example of this event, see "Change Security Equals" on page 802.	DSE_DELETE_VALUE
Security	VERIFY_PASSWORD	DSE_VERIFY_PASS
	For an example of this event, see "Verify Password" on page 802.	
Security	AUDIT_CONFIG	DSE_ADD_VALUE
	For an example of this event, see "Audit Config" on page 802.	DSE_DELETE_VALUE
Security	CHANGE_PASSWORD	DSE_CHGPASS
	For an example of this event, see "Change Password" on page 802.	DSE_NMAS_LOG_SET_PWD
		DSE_NMAS_LOG_SET_DIST_PWD
		DSDSE_NMAS_LOG_DELETE_PWD
		DSE_NMAS_LOG_DELETE_DIST_PWD
		DSE_NMAS_LOG_CHANGE_PWD
		DSE_NMAS_LOG_DELETE_ALL_LOGIN_SECRET
		DSE_NMAS_LOG_SET_LOGIN_SECRET
		DSE_NMAS_LOG_DELETE_LOGIN_SECRET

Event Category	CEF Event	OpenText eDirectory Event
Security	CHANGE_LOGIN_CONFIG For an example of this event, see "Change Login Config" on page 803.	DSE_NMAS_LOG_SET_LOGIN_CONFIG DSE_NMAS_LOG_DELETE_LOGIN_CONFIG DSE_NMAS_LOG_DELETE_ALL_LOGIN_CONFIG
Security	QUERY_CREDENTIALS For an example of this event, see "Query Credentials" on page 803.	DSE_NMAS_LOG_GET_PWD_HISTORY DSE_NMAS_LOG_GET_PWD DSE_NMAS_LOG_GET_LOGIN_CONFIG DSE_NMAS_LOG_CHECK_PWD_SYNTAX_POLICY DSE_NMAS_LOG_GET_ALL_LOGIN_SECRET DSE_NMAS_LOG_GET_PWD_STATUS DSE_NMAS_LOG_GET_ALL_LOGIN_CONFIG DSE_NMAS_LOG_GET_DIST_PWD DSE_NMAS_LOG_GET_PWD_HISTORY
Security	IMPERSONATE For an example of this event, see "Impersonate" on page 803.	DSE_IMPERSONATE
Security	AUTHENTICATE For an example of this event, see "Authenticate" on page 804.	DSE_AUTHENTICATE
Objects	CREATE_OBJECT For an example of this event, see "Create Object" on page 805.	DSE_CREATE_ENTRY DSE_ADD_ENTRY
Objects	DELETE_OBJECT For an example of this event, see "Delete Object" on page 805.	DSE_REMOVE_ENTRY
Objects	RENAME_OBJECT For an example of this event, see "Rename Object" on page 805	DSE_RENAME_ENTRY
Objects	MOVE_OBJECT For an example of this event, see "Move Object" on page 805.	DSE_MOVE_SOURCE_ENTRY DSE_MOVE_DEST_ENTRY
Objects	DSA_READ For an example of this event, see "DSA Read" on page 806.	DSE_DSA_READ

Event Category	CEF Event	OpenText eDirectory Event
Objects	SEARCH	DSE_SEARCH
	For an example of this event, see "Search" on page 806.	
Attributes	READ_ATTRIBUTE	DSE_READ_ATTR
	For an example of this event, see "Read Attribute" on page 807.	
Attributes	DELETE_ATTRIBUTE	DSE_DELETE_ATTRIBUTE
	For an example of this event, see "Delete Attribute" on page 807.	
Attributes	ADD_VALUE	DSE_ADD_VALUE
	For an example of this event, see "Add Value" on page 807.	
Attributes	DELETE_VALUE	DSE_DELETE_VALUE
Attributes	COMPARE_ATTRIBUTE_VALUE	DSE_COMPARE_ATTR_VALUE
	For an example of this event, see "Compare Attribute Value" on page 808.	
LDAP	LDAP_BIND	DSE_LDAP_BIND
	LDAP_UNBIND	DSE_LDAP_UNBIND
	LDAP_CONNECTION	DSE_LDAP_CONNECTION
	LDAP_SEARCH	DSE_LDAP_SEARCH
	LDAP_ADD	DSE_LDAP_ADD
	LDAP_COMPARE	DSE_LDAP_COMPARE
	LDAP_MODIFY	DSE_LDAP_MODIFY
	LDAP_DELETE	DSE_LDAP_DELETE
	LDAP_MODIFY_DN	DSE_LDAP_MODDN
	LDAP_ABANDON	DSE_LDAP_ABANDON
	LDAP_EXTENDED_OPERATION	DSE_LDAP_EXTOP
	LDAP_SYSTEM_EXTENDED_OPERATION	DSE_LDAP_SYSEXTOP
	LDAP_PASSWORD_MODIFY	DSE_LDAP_PASSWDMODIFY
	MODIFY_LDAP_SERVER_CONFIGURATION	DSE_LDAP_MODLDAPSERVER
	UNKNOWN_LDAP_OPERATION	DSE_LDAP_UNKNOWNOP
	LDAP_BIND_RESPONSE	DSE_LDAP_BINDRESPONSE

Event Category	CEF Event	OpenText eDirectory Event
	LDAP_SEARCH_RESPONSE	DSE_LDAP_SEARCHRESPONSE
	LDAP_SEARCH_ENTRY_RESPONSE	DSE_LDAP_SEARCHENTRYRESPONSE
	LDAP_ADD_RESPONSE	DSE_LDAP_ADDRESPONSE
	LDAP_COMPARE_RESPONSE	DSE_LDAP_COMPARERESPONSE
	LDAP_MODIFY_RESPONSE	DSE_LDAP_MODIFYRESPONSE
	LDAP_DELETE_RESPONSE	DSE_LDAP_DELETERESPONSE
	LDAP_MODIFY_DN_RESPONSE	DSE_LDAP_MODDNRESPONSE
	LDAP_EXTENDED_OPERATION_RESPONSE	DSE_LDAP_EXTOP_RESPONSE
EBA	MODIFY_SERVICE_CONFIG	DSE_EBA_ISSUE_NCPCA_CERT
	For an example of this event, see "Modify Service Config" on page 808.	DSE_EBA_REVOKE_NCPCA_CERT
		DSE_EBA_MOVE_EBA_CA
		DSE_EBA_ISSUE_CRL
		DSE_EBA_REQ_SERVER_BA_MATERIAL

CEF Events

The CEF events are classified into the following categories:

- "Security Events" on page 797
- "Objects Events" on page 804
- "Attribute Events" on page 806
- "EBA Events" on page 808

Security Events

This set of events are applicable for auditing security operations of OpenText eDirectory. A security operation may be granting or revoking access, login, password modification or query. This set of events also help to detect intruder attempts on the OpenText eDirectory system.

Examples of Security Events:

This section includes the examples for the following Security Events:

- "Connection" on page 798
- "Login" on page 798
- "Logout" on page 799
- "Add Member" on page 799

- "Delete Member" on page 799
- "Intruder Detected" on page 800
- "Account Unlock" on page 800
- "Login Disabled" on page 800
- "Login Enabled" on page 801
- "ACL Changed" on page 801
- "Change Security Equals" on page 802
- "Verify Password" on page 802
- "Audit Config" on page 802
- "Change Password" on page 802
- "Change Login Config" on page 803
- "Query Credentials" on page 803
- "Impersonate" on page 803
- "Authenticate" on page 804

NOTE: The examples provided in the following sections are for reference only.

Connection

Click **Connection** to generate an event when a communication channel is created between system components, as shown in the following example:

```
Oct 31 17:00:22 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B035E|CONNECTION|0|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Oct 31 2017 17:00:22 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#DS src=164.99.179.164 spt=23017 duser=CN\=SLES12SP2-
194,OU\=server,OU\=co,O\=in cn1Label=Connection ID cn1=246358976
cn2Label=Created(1)/Terminated(0) cn2=1 cs1Label=Client Address
cs1=164.99.179.164:23017 cs2Label=Module cs2=LDAP Server cs3Label=Tree
Name cs3=TEST-CEF-AGN cs4Label=Correlation ID cs4=eDirectory#4294967295#
flexString2Label=SubEvent flexString2=DSE_CONNECTION cat=Security reason=0
outcome=Success
```

Login

Click **Login** to generate an event when a new session is created. For example, logging in to the OpenText eDirectory system.

```
Oct 31 17:00:22 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B035C|LOGIN|1|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Oct 31 2017 17:00:22 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#NMAS src=164.99.179.164 spt=59737
suser=CN\=admin,OU\=novell,OU\=co,O\=in
duser=CN\=admin,OU\=novell,OU\=co,O\=in cs1Label=Client Address
cs1=164.99.179.164:59737 cs2Label=Class Name cs2=User cs3Label=Tree Name
cs3=TEST-CEF-AGN cs4Label=Correlation ID cs4=nmas#262183# cs6Label=Server
Name cs6=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in flexString1Label=Login
Method flexString1=0 flexString2Label=SubEvent
flexString2=DSE_NMAS_LOG_FINISH_LOGIN_STATUS flexNumber2Label=Grouping
flexNumber2=386 cat=Security reason=0 outcome=Success
```

Logout

Click **Logout** to generate an event when an existing session is terminated. For example, logging out of the OpenText eDirectory system.

```
Jan 09 18:34:15 eDirectory
CEF:0|NetIQ|eDirectory|9.1|CEF0B0303|LOGOUT|1|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Nov 03 2017 13:10:32 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#DS src=164.99.44.5 spt=53738 suser=[Public]
duser=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in cs1Label=Client Address
cs1=164.99.44.5 cs2Label=Class Name cs2=User cs3Label=Tree Name cs3=TEST-
CEF-NOV3 cs4Label=Correlation ID cs4=eDirectory#17# cs6Label=Object DN
cs6=CN\=admin,OU\=novell,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_LOGOUT flexNumber2Label=Grouping flexNumber2=127
cat=Security reason=0 outcome=Success
```

Add Member

Click **Add Member** to generate an event when a new user is added to the group, as shown in the following example:

```
Jan 09 18:34:15 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0336|ADD_MEMBER|1|dvc=164.99.179.156
dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:34:15 dtz=IST
sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.158
spt=54936 suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=CN\=grp1,OU\=lnx-
users,OU\=novell,OU\=co,O\=in cs2Label=Class Name cs2=Group cs3Label=Tree
Name cs3=NEW-TREE-9th cs4Label=Correlation ID cs4=eDirectory#14#bc560efc-
53d4-4ad9-85b4-fc0e56bcd453 cs6Label=Member DN cs6=CN\=lynx-user,OU\=lnx-
users,OU\=novell,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_ADD_VALUE flexNumber2Label=Grouping flexNumber2=3676
cat=Security reason=0 outcome=Success
```

Delete Member

Click **Delete Member** to generate an event when a user is removed from the group, as shown in the following example:

```
Jan 09 18:35:06 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0337|DELETE_MEMBER|1|dvc=164.99.179.1
56 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:35:06
dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.158
spt=54936 suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=CN\=grp1,OU\=lnx-
users,OU\=novell,OU\=co,O\=in cs2Label=Class Name cs2=Group cs3Label=Tree
Name cs3=NEW-TREE-9th cs4Label=Correlation ID cs4=eDirectory#14#9136617f-
4412-48da-bf33-7f6136911244 cs6Label=Member DN cs6=CN\=lynx-user,OU\=lnx-
users,OU\=novell,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_DELETE_VALUE flexNumber2Label=Grouping flexNumber2=3687
cat=Security reason=0 outcome=Success
```

Intruder Detected

Click **Intruder Detected** to generate an event when an intruder is detected, as shown in the following example:

```
Jan 09 18:35:06 eDirectory
CEF:0|NetIQ|eDirectory|9.1|CEF0B0357|INTRUDER_DETECTED|5|dvc=164.99.179.19
4 dvchost=SLES12SP2-194 rt=Oct 17 2017 19:50:20 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#DS src=164.99.179.194 spt=0 suser=CN\=SLES12SP2-
194,OU\=server,OU\=co,O\=in
duser=CN\=raghu,OU\=lens,OU\=QA,OU\=HD,OU\=DSLR,OU\=SLR,OU\=digital,OU\=ca
mera,O\=sony,L\=tokyo,dc\=co,C\=jp cs1Label=Intruder Address cs1=TCP:
164.99.179.164:33584 cs2Label=Reset Time cs2=10/17/17 19:52:20
cs3Label=Tree Name cs3=TEST-CEF222 cs4Label=Correlation ID
cs4=eDirectory#0#349e5670-0b80-4c99-b7f0-70569e34800b cs6Label=Class
cs6=User flexString2Label=SubEvent flexString2=DSE_ADD_VALUE
flexNumber2Label=Grouping flexNumber2=102 cat=Security reason=0
outcome=Success
```

Account Unlock

Click **Account Unlock** to generate an event when a locked account is unlocked, as shown in the following example:

```
Jan 09 19:10:32 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B035F|ACCOUNT_UNLOCK|2|dvc=164.99.179.
156 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 19:10:32
dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.156
spt=0 suser=CN\=SLES12-SP3-156,OU\=lnx-server,OU\=server,OU\=co,O\=in
duser=CN\=rr,OU\=lnx-users,OU\=novell,OU\=co,O\=in cs2Label=Class Name
cs2=User cs3Label=Tree Name cs3=NEW-TREE-9th cs4Label=Correlation ID
cs4=eDirectory#0#ad3a0226-764e-488c-b90a-26023aad4e76
flexString2Label=SubEvent flexString2=DSE_DELETE_VALUE
flexNumber2Label=Grouping flexNumber2=122 cat=Security reason=0
outcome=Success
```

Login Disabled

Click **Login Disabled** to generate an event when a user account is disabled, as shown in the following example:

```
Jan 09 18:18:48 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0356|LOGIN_DISABLED|2|dvc=164.99.179.
156 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:18:48
dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.158
spt=54936 suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=CN\=lynx-
user1,OU\=lnx-users,OU\=novell,OU\=co,O\=in cs2Label=Class Name cs2=User
cs3Label=Tree Name cs3=NEW-TREE-9th cs4Label=Correlation ID
cs4=eDirectory#14#f04b6deb-df9b-4f4b-a8e8-eb6d4bf09bdf
flexString2Label=SubEvent flexString2=DSE_ADD_VALUE
flexNumber2Label=Grouping flexNumber2=100 cat=Security reason=0
outcome=Success
```

Login Enabled

Click **Login Enabled** to generate an event when a disabled user account is enabled, as shown in the following example:

```
Jan 09 18:18:56 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0355|LOGIN_ENABLED|2|dvc=164.99.179.1
56 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:18:56
dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.158
spt=54936 suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=CN\=lynx-
user1,OU\=lnx-users,OU\=novell,OU\=co,O\=in cs2Label=Class Name cs2=User
cs3Label=Tree Name cs3=NEW-TREE-9th cs4Label=Correlation ID
cs4=eDirectory#14#f99f0883-251e-424e-a724-83089ff91e25
flexString2Label=SubEvent flexString2=DSE_DELETE_VALUE
flexNumber2Label=Grouping flexNumber2=107 cat=Security reason=0
outcome=Success
```

ACL Changed

Click **ACL Changed** to generate an event when an ACL is changed on an object, as shown in the following example:

```
Jan 09 18:04:56 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0354|ACL_CHANGED|3|dvc=164.99.179.156
dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:04:56 dtz=IST
sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.158
spt=52120 suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=CN\=lynx-
user,OU\=lnx-users,OU\=novell,OU\=co,O\=in cn1Label=ACL Added cn1=1
cs1Label=Value cs1=Entry ID: .CN\=lynx-user.OU\=lnx-
users.OU\=novell.OU\=co.O\=in.T\=NEW-TREE-9th., Attribute ID: [All
Attributes Rights], Privileges: Attribute Read cs2Label=Class Name cs2=User
cs3Label=Tree Name cs3=NEW-TREE-9th cs4Label=Correlation ID
cs4=eDirectory#18#c4f344f7-db17-4366-8a19-f744f3c417db cs6Label=Trustee
cs6=CN\=lynx-user,OU\=lnx-users,OU\=novell,OU\=co,O\=in
flexString2Label=SubEvent flexString2=DSE_ADD_VALUE
flexNumber2Label=Grouping flexNumber2=83 cat=Security reason=0
outcome=Success
```

Change Security Equals

Click **Change Security Equals** to generate an event when Security Equals is changed on an object, as shown in the following example:

```
Jan 09 18:29:38 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0341|CHANGE_SECURITY_EQUALS|3|dvc=164
.99.179.156 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018
18:29:38 dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.156
spt=0 suser=CN\=SLES12-SP3-156,OU\=lnx-server,OU\=server,OU\=co,O\=in
duser=CN\=raghu,OU\=lnx-users,OU\=novell,OU\=co,O\=in cn1Label=Add/Remove
cn1=1 cs2Label=Class Name cs2=User cs3Label=Tree Name cs3=NEW-TREE-9th
cs4Label=Correlation ID cs4=eDirectory#0#6d1355d0-0401-4858-8475-
d055136d0104 cs6Label=Equivalent DN cs6=CN\=grp,OU\=novell,OU\=co,O\=in
flexString2Label=SubEvent flexString2=DSE_ADD_VALUE
flexNumber2Label=Grouping flexNumber2=3639 cat=Security reason=0
outcome=Success
```

Verify Password

Click **Verify Password** to generate an event when an account password is verified.

Audit Config

Click **Audit Config** to generate an event when any modification is done to the parameters that are controlling the audit service, as shown in the following example:

```
Jan 09 18:27:12 eDirectory
CEF:0|eDirectory|eDirectory|9.1|CEF0B0006|AUDIT_CONFIG|2|dvc=164.99.179.15
6 dvchost=SLES12-SP3-156.labs.blr.novell.com rt=Jan 09 2018 18:27:12
dtz=IST sourceServiceName=CN\=SLES12-SP3-156,OU\=lnx-
server,OU\=server,OU\=co,O\=in sproc=eDirectory#DS src=164.99.179.160
spt=54980 suser=CN\=srv-160,OU\=server,OU\=co,O\=in duser=CN\=SLES12-SP3-
156,OU\=lnx-server,OU\=server,OU\=co,O\=in cs1Label=Attribute Value
cs1=cefEvents\=ACL_CHANGED $$QUERY_CREDENTIALS cs2Label=Class Name cs2=NCP
Server cs3Label=Tree Name cs3=NEW-TREE-9th cs4Label=Correlation ID
cs4=eDirectory#16#8dcd3ede-baf8-4e71-9f1e-de3ecd8df8ba cs6Label=Attribute
Name cs6=cefConfiguration flexString2Label=SubEvent
flexString2=DSE_ADD_VALUE flexNumber2Label=Grouping flexNumber2=3631
cat=Security reason=0 outcome=Success
```

Change Password

Click **Change Password** to generate an event when an account password is changed, as shown in the following example:

```
Oct 31 17:06:11 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0290064|CHANGE_PASSWORD|1|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Oct 31 2017 17:06:11 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#NMAS src=164.99.179.194 spt=0
suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=raghu,novell,co,in
cs2Label=Class Name cs2=User cs3Label=Tree Name cs3=TEST-CEF-AGN
cs4Label=Correlation ID cs4=nmas#0# cs6Label=Server Name
cs6=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_NMAS_LOG_SET_LOGIN_SECRET flexNumber2Label=Grouping
flexNumber2=405 cat=Security reason=0 outcome=Success
```

Change Login Config

Click **Change Login Config** to generate an event when an account login configuration is changed, as shown in the following example:

```
Nov 02 10:21:00 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0290061|CHANGE_LOGIN_CONFIG|1|dvc=164.99.179
.194 dvchost=SLES12SP2-194 rt=Nov 02 2017 10:21:00 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#NMAS src=164.99.179.194 spt=0
suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=raghu,novell,co,in
cs2Label=Class Name cs2=User cs3Label=Tree Name cs3=TEST-CEF-AGN
cs4Label=Correlation ID cs4=nmas#0# cs6Label=Server Name
cs6=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_NMAS_LOG_SET_LOGIN_CONFIG flexNumber2Label=Grouping
flexNumber2=2034 cat=Security reason=0 outcome=Success
```

Query Credentials

Click **Query Credentials** to generate an event whenever a request for the credential information of a particular account is made, as shown in the following example:

```
Nov 02 10:21:00 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0290062|QUERY_CREDENTIALS|1|dvc=164.99.179.1
94 dvchost=SLES12SP2-194 rt=Nov 02 2017 10:21:00 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#NMAS src=164.99.179.194 spt=0
suser=CN\=admin,OU\=novell,OU\=co,O\=in duser=raghu,novell,co,in
cs2Label=Class Name cs2=User cs3Label=Tree Name cs3=TEST-CEF-AGN
cs4Label=Correlation ID cs4=nmas#0# cs6Label=Server Name
cs6=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_NMAS_LOG_GET_LOGIN_CONFIG flexNumber2Label=Grouping
flexNumber2=2035 cat=Security reason=0 outcome=Success
```

Impersonate

Click **Impersonate** to generate an event whenever an impersonation of an account takes place, as shown in the following example:

```
Nov 02 10:29:38 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0231|IMPERSONATE|1|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Nov 02 2017 10:29:38 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#DS src=164.99.179.194 spt=56451
suser=CN\=admin,OU\=novell,OU\=co,O\=in
duser=CN\=raghu,OU\=novell,OU\=co,O\=in cs3Label=Tree Name cs3=TEST-CEF-
AGN cs4Label=Correlation ID cs4=eDirectory#10# cs6=CN\=SLES12SP2-
194,OU\=server,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_IMPERSONATE flexNumber2Label=Grouping flexNumber2=2048
cat=Security reason=0 outcome=Success
```

Authenticate

Click **Authenticate** to generate an event when a user authenticates a session, as shown in the following example:

```
Nov 02 10:32:39 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B035D|AUTHENTICATE|1|dvc=164.99.179.194
dvchost=SLES12SP2-194 rt=Nov 02 2017 10:32:39 dtz=IST
sourceServiceName=CN\=SLES12SP2-194,OU\=server,OU\=co,O\=in
sproc=eDirectory#DS src=164.99.179.194 spt=0
suser=CN\=impuser,OU\=novell,OU\=co,O\=in
duser=CN\=impuser,OU\=novell,OU\=co,O\=in cs2Label=Class Name cs2=User
cs3Label=Tree Name cs3=TEST-CEF-AGN cs4Label=Correlation ID
cs4=eDirectory#12# cs6Label=Server Name cs6=CN\=SLES12SP2-
194,OU\=server,OU\=co,O\=in flexString2Label=SubEvent
flexString2=DSE_AUTHENTICATE flexNumber2Label=Grouping flexNumber2=2058
cat=Security reason=0 outcome=Success
```

Objects Events

This set of events are applicable for auditing object related operations of OpenText eDirectory. An object operation may be creating, deleting, renaming, moving or querying objects.

Examples of Objects Events:

This section includes the examples for the following Objects Events:

- “Create Object” on page 805
- “Delete Object” on page 805
- “Rename Object” on page 805
- “Move Object” on page 805
- “DSA Read” on page 806
- “Search” on page 806

NOTE: The examples provided in the following sections are for reference only.

Create Object

Click **Create Object** to generate an event when a new object is created in the OpenText eDirectory tree, as shown in the following example:

```
Oct 23 23:57:19 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0001|CREATE_OBJECT|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 23 2017 23:57:19 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.58 spt=52362 suser=CN\=Admin,O\=novell
duser=CN\=user001,O\=novell cs2Label=Class Name cs2=User cs3Label=Tree
Name cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#17#dc0fee11-5cd9-
47d4-b981-cdb8ecd47e07 flexString2Label=SubEvent
flexString2=DSE_CREATE_ENTRY flexNumber2Label=Grouping flexNumber2=677768
cat=Objects reason=0 outcome=Success
```

Delete Object

Click **Delete Object** to generate an event when an object is removed from the OpenText eDirectory tree, as shown in the following example:

```
Oct 24 00:02:35 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0309|DELETE_OBJECT|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 00:02:35 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.58 spt=52362 suser=CN\=Admin,O\=novell
duser=CN\=user001,O\=novell cs2Label=Class Name cs2=User cs3Label=Tree
Name cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#17#2b97f69d-2984-
4f96-a83c-0b6c828bc462 flexString2Label=SubEvent
flexString2=DSE_REMOVE_ENTRY flexNumber2Label=Grouping flexNumber2=677993
cat=Objects reason=0 outcome=Success
```

Rename Object

Click **Rename Object** to generate an event when an object is renamed, as shown in the following example:

```
Oct 24 02:06:23 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0003|RENAME_OBJECT|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:06:23 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.58 spt=55434 suser=CN\=Admin,O\=novell
duser=CN\=u1,O\=novell cs2Label=Class Name cs2=User cs3Label=Tree Name
cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#17#28250918-af9c-4098-
b56a-5757e456102a cs6Label=New Object DN cs6=CN\=u1changed,O\=novell
flexString2Label=SubEvent flexString2=DSE_RENAME_ENTRY
flexNumber2Label=Grouping flexNumber2=683314 cat=Objects reason=0
outcome=Success
```

Move Object

Click **Move Object** to generate an event when an object is moved, as shown in the following example:

```
Oct 24 02:18:57 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0004|MOVE_OBJECT|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:18:57 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.58 spt=55434 suser=CN\=Admin,O\=novell
duser=CN\=u1changed,O\=novell cs2Label=Class Name cs2=User cs3Label=Tree
Name cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#17#28789395-394f-
49d5-bb4e-b95410b0f9b5 cs6Label=New DN cs6=CN\=u1changed,OU\=org,O\=novell
flexString2Label=SubEvent flexString2=DSE_MOVE_SOURCE_ENTRY
flexNumber2Label=Grouping flexNumber2=683861 cat=Objects reason=0
outcome=Success
```

DSA Read

Click **DSA Read** to generate an event when an object is read, as shown in the following example:

```
Oct 24 02:36:27 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0230|DSA_READ|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:36:27 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=20928 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell cs2Label=Class Name cs2=NCP Server
cs3Label=Tree Name cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#1#
flexString2Label=SubEvent flexString2=DSE_DSA_READ cat=Objects reason=0
outcome=Success
```

Search

Click **Search** to generate an event when a request for a search operation is made, as shown in the following example:

```
Oct 24 02:36:29 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B033C|SEARCH|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:36:29 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=21184 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=CN\=Security cn1Label=Scope cn1=2 cn2Label=Nodes To Search cn2=100
cs2Label=Class Name cs2=SAS:Security cs3Label=Tree Name cs3=TREE910W
cs4Label=Correlation ID cs4=eDirectory#2# flexString2Label=SubEvent
flexString2=DSE_SEARCH flexNumber2Label=Grouping flexNumber2=684639
cat=Objects reason=0 outcome=Success
```

Attribute Events

This set of events are applicable for auditing attribute related operations of OpenText eDirectory. An attribute operation may be creating, deleting, renaming, moving or searching attribute.

Examples of Attribute Events:

This section includes the examples for the following Attribute Events:

- “Read Attribute” on page 807
- “Delete Attribute” on page 807

- "Add Value" on page 807
- "Delete Value" on page 808
- "Compare Attribute Value" on page 808

NOTE: The examples provided in the following sections are for reference only.

Read Attribute

Click **Read Attribute** to generate an event when an attribute is read on an object, as shown in the following example:

```
Oct 26 11:38:35 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0323|READ_ATTRIBUTE|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 25 2017 23:08:35 dtz=India Standard Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=18369 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell cs2Label=Class Name cs2=NCP Server
cs3Label=Tree Name cs3=TREE910W cs4Label=Correlation ID cs4=eDirectory#1#
cs6Label=Attribute Name cs6=cefConfiguration flexString2Label=SubEvent
flexString2=DSE_READ_ATTR cat=Attributes reason=0 outcome=Success
```

Delete Attribute

Click **Delete Attribute** to generate an event when an attribute is removed from an object, as shown in the following example:

```
Oct 24 22:54:36 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0009|DELETE_ATTRIBUTE|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 22:54:36 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=21184 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell cs2Label=Class Name cs2=NCP Server
cs3Label=Tree Name cs3=TREE910W cs4Label=Correlation ID
cs4=eDirectory#2#a9ea8944-6a78-4a69-9c11-727635aa79e8 cs6Label=Attribute
Name cs6=Network Address flexString2Label=SubEvent
flexString2=DSE_DELETE_ATTRIBUTE flexNumber2Label=Grouping
flexNumber2=736694 cat=Attributes reason=0 outcome=Success
```

Add Value

Click **Add Value** to generate an event when a value is added to an attribute, as shown in the following example:

```
Oct 24 02:38:12 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0006|ADD_VALUE|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:38:12 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=0 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=.[Pseudo Server] cs1Label=Attribute Value cs1=720575940530274304
cs3Label=Tree Name cs3=TREE910W cs4Label=Correlation ID
cs4=eDirectory#0#f9787bd7-0541-47ca-9391-5a4bada90f02 cs6Label=Attribute
Name cs6=treeReferral flexString2Label=SubEvent flexString2=DSE_ADD_VALUE
flexNumber2Label=Grouping flexNumber2=684713 cat=Attributes reason=0
outcome=Success
```

Delete Value

Click **Delete Value** to generate an event when a value is removed from an attribute, as shown in the following example:

```
Oct 24 02:38:12 NetIQ
CEF:0|NetIQ|eDirectory|9.1|CEF0B0007|DELETE_VALUE|0|dvc=164.99.179.60
dvchost=WIN-37D8M9SKD2U rt=Oct 24 2017 02:38:12 dtz=Pacific Daylight Time
sourceServiceName=CN\=WIN-37D8M9SKD2U-NDS,O\=novell sproc=eDirectory#DS
src=164.99.179.60 spt=0 suser=CN\=WIN-37D8M9SKD2U-NDS,O\=novell
duser=.[Pseudo Server] cs1Label=Attribute Value cs1=720575940530274304
cs3Label=Tree Name cs3=TREE910W cs4Label=Correlation ID
cs4=eDirectory#0#1c411e7f-9657-474e-8e8e-80fc92921f96 cs6Label=Attribute
Name cs6=localReferral flexString2Label=SubEvent
flexString2=DSE_DELETE_VALUE flexNumber2Label=Grouping flexNumber2=684714
cat=Attributes reason=0 outcome=Success
```

Compare Attribute Value

Click **Compare Attribute Value** to generate an event when an attribute value is compared, as shown in the following example:

EBA Events

This set of events are applicable for auditing EBA related operations of OpenText eDirectory. EBA related operations may be enabling or disabling EBA, server addition or removal, movement of EBACA from one server to another or even `ndslogin`.

Examples of EBA Events:

This section includes the examples for the following EBA Events:

- "Modify Service Config" on page 808

NOTE: The examples provided in the following sections are for reference only.

Modify Service Config

Click **Modify Service Config** to generate an event when there are EBA related changes in the OpenText eDirectory tree, as shown in the following example:

```
Sep 17 16:34:56 eDirectory
CEF:0|eDirectory|eDirectory|9.2|CEF0B0503|MODIFY_SERVICE_CONFIG|1|dvc=164.
99.179.216 dvchost=SLESSP1-216 rt=Sep 17 2019 16:34:56 dtz=IST
sourceServiceName=CN\=SLESSP1-216,O\=novell sproc=eDirectory#DS
src=164.99.179.216 spt=0 cs1Label=CertAuthOldSvrAddress cs1=164.99.179.216
cs2Label=CertAuthNewSvrAddress cs2=164.99.179.216 cs3Label=Tree Name
cs3=TREE216 cs4Label=Correlation ID cs4=eDirectory#4294967295#
cs6Label=Server Name cs6=CN\=SLESSP1-216,O\=novell
flexString2Label=SubEvent flexString2=DSE_EBA_MOVE_EBA_CA
flexNumber2Label=Grouping flexNumber2=205 cat=Security reason=0
outcome=Success
```

H Troubleshooting

- “Troubleshooting SNMP” on page 811
- “Troubleshooting OpenText iMonitor” on page 815
- “Troubleshooting Obituaries” on page 817
- “Migrating to OpenText eDirectory” on page 820
- “Troubleshooting Schema” on page 826
- “Troubleshooting DSRepair” on page 827
- “Troubleshooting Replication” on page 827
- “Troubleshooting Clone DIB Issues” on page 828
- “Troubleshooting OpenText Public Key Infrastructure Services” on page 828
- “Troubleshooting Utilities on Linux” on page 834
- “Troubleshooting NMAS” on page 835
- “Accessing HTTPSTK When Directory Service Is Not Loaded” on page 837
- “Troubleshooting Data Encryption” on page 839
- “The OpenText eDirectory Management Toolbox” on page 842
- “Troubleshooting Issues with SASL-GSSAPI” on page 843
- “Managing Error Logging in OpenText eDirectory” on page 845
- “Miscellaneous” on page 848
- “Troubleshooting IPV6 Issues” on page 856
- “Troubleshooting EBA” on page 856

Troubleshooting SNMP

Traps Might Not Get Generated As Expected

Traps are sent only if the corresponding verb request is received by the server. They are not sent in any other cases. For example, `ndsDeleteAttribute` is sent only when the `ndsRemoveEntry` (trap number 108) request is sent. But an application can always read the ACLs and decide to check whether the user has sufficient rights to perform the delete operation. In this case, the `ndsDeleteAttribute` trap is not generated. However, you can use OpenText iMonitor to view the verb statistics on a particular server.

To get the traps for all occurrences, set the time interval to zero.

You can enable traps to send only on failure conditions. You can enable traps to get them under all conditions.

ndssnmpsa must be restarted when the master agent is restarted

To restart ndssnmpsa, stop ndssnmpsa and then start it again.

To stop ndssnmpsa, enter the following:

Linux: `/etc/init.d/ndssnmpsa stop`

To start ndssnmpsa, enter the following:

Linux: `/etc/init.d/ndssnmpsa start`

SNMP Group Object

If the installation of the SNMP Group object fails, you can rectify this problem by executing the following command on the server console:

`ndsconfig add -m snmp`

SNMP Object Creation Error on Windows Server

While installing OpenText eDirectory on any supported Windows platform server, if you get an SNMP group object creation error, you need to manually create the SNMP group object. For information on the steps to manually create an SNMP object, see Chapter 18, “SNMP Support for OpenText eDirectory,” on page 489.

OpenText eDirectory SNMP initialization component. Error code: -255

or

Initialization failure. Error code: -255

The possible cause could be that you have not specified `hostname:port` or `IP_address:port` as a paramater to the SERVER command in OpenText eDirectory SNMP configuration file.

The OpenText eDirectory SNMP configuration file is `ndssnmp.cfg`. It is located in the following directories:

- Linux: `/etc/opt/novell/eDirectory/conf/ndssnmp/`
- Windows: *`install_directory`*`\SNMP\`

LDAP SNMP Statistics Not Report

When anonymous bind is disabled, LDAP SNMP statistics are not reported.

To resolve this issue:

1. Allow anonymous bind.
2. Start the subagent.
3. Disable/disallow anonymous bind.

Segmentation Fault Error while Accessing the Subagent

When a user tries to start the subagent (`ndssnmpsa`) by using an incorrect OpenText eDirectory password, a segmentation fault error occurs.

To avoid getting this error, ensure that you use the correct OpenText eDirectory password while starting the subagent.

Issues After Upgrading from OpenText eDirectory 8.7.3 to OpenText eDirectory 9.0

After upgrading from eOpenText Directory 8.7.3 to OpenText eDirectory 9.0, you might get the following error:

```
%%% Attempting to restart the NetIQ eDirectory SNMP subagent (ndssnmpsa)...
Starting NDS SNMP Subagent ...
Initialization failure. Error code : -255
Please Wait...
Done

%%% Unable to start ndssnmpsa... Please try starting it manually...
```

This error occurs because with OpenText eDirectory 9.0, OpenText eDirectory does not listen on the localhost. Earlier the `ndssnmp.cfg` file had `SERVER localhost` set by default.

To resolve this error, you need to manually edit the `ndssnmp.cfg` file and include the host name of the OpenText eDirectory server, which needs to be monitored.

For example, type the following in the `ndssnmp.cfg` file:

```
SERVER test-server
```

test-server is the hostname on which OpenText eDirectory is running on the default NCP port (that is 524). If OpenText eDirectory is running on a different port (for ex: 1524), the entry should be as follows:

```
SERVER test-server:1524
```

Errors While Starting the NDS Subagent

The subagent can fail with the following message:

```
Unable to load library: libnetsnmp.so
```

To resolve this, export the environment variable SNMP_MAJOR_VERSION with the net-snmp library's (`libnetsnmp.so`) major version number. For example, you might use the following command:

```
export SNMP_MAJOR_VERSION=10
```

Restarting ndssnmpsa

When the master agent is restarted on Linux, ndssnmpsa needs to be restarted.

To restart ndssnmpsa, stop ndssnmpsa and then start it again.

To stop ndssnmpsa, enter the following command:

```
/etc/init.d/ndssnmpsa stop
```

To start ndssnmpsa, enter the following:

```
/etc/init.d/ndssnmpsa start
```

Compiling edir.mib

The OpenText eDirectory MIB file (<*eDirectoryInstallRootDir*>\snmp\edir.mib) on Windows compiles with some errors and warnings on HP OpenView. You can ignore these errors.

Modifying the SNMP Configuration File

If LDAP is not configured to run in clear text mode, the name of the trusted root certificate file must be given in the SNMP configuration file (for example, SSLKEY C:\Novell\nds\trust.der) before bringing up the OpenText eDirectory SNMP subagent.

ndssnmp.cfg is found in C:\novell\nds\snmp on Windows.

Using SNMP After a New Tree Installation

When you install OpenText eDirectory 9.0 for the first time (creating a new tree), if the Windows SNMP Service is installed on the server, and the SNMP Service has one or more dependent services, OpenText eDirectory cannot shut down the SNMP Service. If this happens, SNMP is not ready to use after the OpenText eDirectory installation.

Follow these steps to restart the SNMP service:

1. Click **Start > Settings > Control Panel > Administrative Tools > Services**.
2. Right-click **SNMP Service** in the **Name** list, then click **Stop**.
3. Click **Yes to All**.
4. Right-click **SNMP Service** in the **Name** list, then click **Start**.

Uninstalling SNMP with OpenText eDirectory Uninstallation

If the Windows SNMP Service is installed on a server, and the SNMP Service has one or more dependent services, the OpenText eDirectory uninstall does not delete all the SNMP files in the C:\novell\nds folder. However, the other uninstallation processes complete successfully, including the deletion of the SNMP registry entries, and the deconfiguration process that the OpenText SNMP agent does with DS and the SNMP Service.

To complete the uninstallation:

1. Click **Start > Settings > Control Panel > Administrative Tools > Services**.
2. Right-click **SNMP Service** in the **Name** list, then click **Stop**.
3. Click **Yes to All**.
4. Right-click **SNMP Service** in the **Name** list, then click **Start**.
5. Manually delete the remaining SNMP files in the C:\novell\nds folder.

Installing OpenText eDirectory Stops SNMP on Windows 2012

SNMP stops working after installing OpenText eDirectory and displays the following error message:

```
SNMP subagent error -672
```

Workaround:

1. Install and configure SNMP service after OpenText eDirectory is installed.
2. Run the `dssnmpsupport.exe` on your OpenText eDirectory server.

NOTE: Apply `dssnmpsupport.exe` only if `MpsSvc service` is running on the OpenText eDirectory server.

SNMP Subagent Fails to Load when EBA is Enabled

On a server where EBA is enabled, the SNMP agent fails to load. This happens after entering the command to load the SNMP agent (`/etc/init.d/ndssnmpsa start`) and providing the credentials. The error message returned depends on the value defined for the SERVER parameter in `ndssnmp.cfg`.

To workaround the issue:

- Disable FIPS mode on your OpenText eDirectory server by setting `n4u.server.fips_tls` to `0`.
- Restart OpenText eDirectory.

The SERVER parameter must point to the server's hostname or IP address. The SERVER parameter is located in the `ndssnmp.cfg` file, which is found in `/etc/opt/novell/eDirectory/conf/ndssnmp` directory. By default, it is set to `localhost`, but it should be changed to either the IP address or hostname of the OpenText eDirectory server.

Troubleshooting OpenText iMonitor

Browsing for Objects Containing Double-Byte Characters in iMonitor

When using OpenText iMonitor to browse an OpenText eDirectory tree for objects, an object with double-byte characters in the name might not correctly hyperlink to the object properties.

Agent Health Check on a Single-Server Tree

The Agent Health check feature in OpenText iMonitor shows a Warning icon in the Results column when run on a single server tree because of the Perishable Data status. This does not mean that the tree is not healthy or that the Agent Health check is not working as designed. Perishable Data indicates the amount of data that has not yet been synchronized to at least one replica. A single server tree, by its nature, means that the data is always at risk for catastrophic failure because there is no other place that the data is replicated. If you lose the hard disk, you lose the data.

If you don't want to view health check warnings about Perishable Data or Readable Replica Counts on your single server tree, you can turn off these health checks by editing the `ndsimonhealth.ini` file to change the following entries:

`perishable_data-active: OFF`

and

`ring_readable-Min_Marginal: 1` or `ring_readable-active: OFF`

This turns off the warnings for Readable Replica Count and Perishable Data.

OpenText iMonitor Report Does Not Save the Records for Each Hour

The custom reports feature in OpenText iMonitor is designed to place the URL specified by the user into the saved report (the saved HTML file) when the custom report is created. That means that when you open a saved custom report that has been run, you see the live (current) data instead of the data captured by the URL at the time the custom report is run. This issue will be resolved in a future release of OpenText iMonitor.

Creation and Modification Time Stamps

Because Linux platforms do not maintain the creation time of a file, OpenText iMonitor shows both the creation and modification times to be the same.

Run Report Screen Layout Not Aligned on OpenText iMonitor

The navigation and assistant frames appear twice on Linux.

To work around this problem, refresh the page.

OpenText iMonitor Displays Error -672

- **Linux:** OpenText iMonitor displays error -672 if dsdump tool is running in parallel with OpenText iMonitor. To resolve this issue, exit the dsdump tool, before starting iMonitor.
- **Windows:** OpenText iMonitor displays error -672 if the dsbrowse or dsedit tool is running in parallel with OpenText iMonitor. To resolve this issue, exit the dsbrowse and dsedit tool before starting OpenText iMonitor.

OpenText iMonitor Displays Error -702

If you assign a Group Entry ACL rights to an user object, OpenText iMonitor displays an error message while validating the entry after upgrading the eDirectory server.

You must manually update the value for `ValidACLFlags` in the `ndsimonhealth.conf` file and restart the OpenText eDirectory server.

Time Stamps Displayed in Hexadecimal Format

If you set a Time syntax attribute with a value prior to January 1, 1970, OpenText iMonitor displays the time stamp for the attribute in hexadecimal format instead of in standard date/time format. OpenText iMonitor displays all attributes with values after January 1, 1970, in date/time format.

Issue with OpenText iMonitor Trace Configuration in Internet Explorer 11

Trace configuration in OpenText iMonitor does not work in Internet Explorer 10.

To work around this issue, launch Internet Explorer 10 in compatibility mode and OpenText iMonitor address to the list of Trusted Sites, then restart the browser.

Troubleshooting Obituaries

Obituaries serve as operational attributes that OpenText eDirectory places on objects to ensure referential integrity during operations such as delete, move, rename, and restore. For example, if Group A has a member, User B, and User B is deleted, the directory automatically removes the reference to User B from Group A. In OpenText eDirectory 9.0, the obituaries generated by the Delete, Move, and Rename operations are optimized by default.

NOTE: Objects with obituaries are considered every time an agent outbound synchronizes, and by the obituary process, which is scheduled to run at the end of an inbound synchronization cycle.

There are three general classifications for obituaries:

- Primary obituaries include the types Dead (0001), Restored (0000), Moved (0002), New RDN (0005), and Tree New RDN (0008).
- Secondary obituaries are generally associated with a Primary obituary and represent the agents and partitions that need to be notified of the operation specified in the Primary obituary. They include the types Back Link (0006), Used By (000C), and Move Tree (000a).
- Tracking obituaries include the types Inhibit Move (0003), Old RDN (0004), and Tree Old RDN (0007).

Obituaries, with the exception of Tracking obituaries, must move through a set of synchronizing states:

- Initial State or Issued (0)
- Notified (1)
- OK to Purge (2)
- Purgeable (4)

The states are recorded in the Flags field in the obituary attribute. Before an obituary can move to the next state, the current state must have been synchronized to all replicas of the real object. In order to determine whether all replicas in the ring have seen a given obituary state, a vector is computed from the transitive vector. In OpenText eDirectory 8.6 and later, a non-stored Obituary Vector is used. In previous versions of OpenText eDirectory, the Purge Vector is used. If the Modification Timestamp (MTS) on the obituary is older than the computed vector, the server responsible for that obituary can advance it to the next state.

For a Secondary obituary of type Back Link, the agent that holds the master replica of the object with the obituary is responsible for advancing the states. For a Secondary obituary of type Used By, the replica agent that created it is responsible for advancing the obituary states as long as that replica still exists. If it does not still exist, the agent holding the master of that partition takes over advancing the obituary states for the Used By obituary. For a Move Tree obituary, the master of the root partition is responsible for advancing the states.

Primary obituaries can be advanced in their states only after all Secondary obituaries have advanced through all of their states. After the Primary obituary reaches its last state, and that state synchronizes to all servers in the ring, all that remains is the object husk, which is an object without attributes—one which can subsequently be purged from the system by the Purge Process. Tracking obituaries are removed after the Primary obituary is ready to be removed or, in the case of `Inhibit_move`, the Tracking obituary is removed after the Primary obituary has moved to the `OBF_NOTIFIED` state on the master replica.

The replica responsible for processing obituaries does so on a background process (the Obituary Process), which is scheduled on a per-partition basis after a given partition finishes an inbound synchronization cycle. If there are no other replicas of the partition, the Outbound Replication Process is still scheduled on the heartbeat interval. The Outbound Replication Process then starts the Obituary Process. The Obituary Process cannot be manually scheduled, nor does it need to be. As synchronization occurs, the transitive vectors are updated, thus advancing the Purge Vector and Obit Vector. As these vectors move forward, the obituary states are allowed to move forward. This, together with the automatic scheduling done upon inbound synchronization, completes the obituary processing cycle. Therefore, the lifeblood of obituary processing is object synchronization.

For an object that is being removed, after all obituaries whose associated Primary obituary is of type Dead have been advanced to the last state (Purgeable), and that state has been synchronized to all replicas, a new process is responsible for removing the remaining entry husk from the database. The Purge Process runs automatically to remove these husks. You can manually schedule the Purge Process and modify its automatic schedule interval in "Viewing Agent Activity" on page 234.

Resolving Orphaned Obituaries

While looking at obituary objects, walk around the replica ring, comparing the obituary around the ring.

- If not all replicas have a copy of the obituary and all attribute values are not purgeable, this object is inconsistent around the replica ring—and this is a case of an orphaned obituary.
- If the object exists on all replicas and is consistent, then it might not be advancing because of synchronization errors, or the obituary process might be getting errors.

To work around this issue:

- **Preferred method:** If OpenText eDirectory 8.6 or later is on any of the servers in the replica ring, browse to the object in iMonitor, then select Send Single Entry. This will perform a nonauthoritative send to all other replicas.
- **Far less desirable method:** If all servers in the replica ring that have a copy of the orphaned obituary are older than OpenText eDirectory 8.6, load DSBrowse with the -a option, browse to the object, then time-stamp the entry. This will make the object as it exists on this server the authoritative copy. We do not recommend making objects authoritative as a matter of practice.

Resolving Orphaned Obituaries on Extrefs

If the obituary is for an object not stored on this server (that is, the object is an External Reference):

- Check to see if the real object has a matching obituary. If not, this obituary has been orphaned.
- If the real object has a matching obituary, troubleshoot and resolve obituary problems on the real object before attempting to address any issues with the obit on the `ExtRef` partition.

To work around this issue:

- **Less desirable method:** Run DSRepair with the time stamp option selected.
- **Less desirable method:** Move a real replica to the server, wait for it to turn on, then wait for the obituary to be processed. After the obituary has processed, the replica can be removed if desired.

Resolving Synchronization Issues with Obituaries

To make sure that the obituaries are correctly synchronized:

- Use the iMonitor Agent Synchronization page to check for and resolve any synchronization errors.
- Obituaries can change states only after all agents holding a copy of the replica ring have seen the state change. There are several ways to ensure that every replica has seen the data:

  While browsing the entry with obituaries, click the Entry Synchronization link. The page displayed will show all attributes that have not been synchronized to all replicas.

  Find the oldest time stamp on any of the obituary attribute values. The difference between that time and the current time should be greater than the interval shown in the Max Ring Delta field on the Partition Synchronization page.

  Evaluate the transitive vector.

Looking for Errors with Obituaries

Examine the Agent Process Status: Obituaries to look for any errors.

- Common problems in Agent Process Status: Obituaries include

  -625, -622, -634, and -635 communication problems. See Server Information Report for more details.

  -601, and -603, indicating servers that have been improperly removed, or that the Server object might have a base class of Unknown.
- Errors shown on this page are not fatal. The next time the obituary process runs for that partition, it will retry the operation. Resolve any issues shown in this page, then wait for the retry.

Previous Practices

In the past, several different strategies have been employed to resolve stuck obituaries. Some of these strategies involve expensive partitioning operations, or the use of undocumented features that might cause problems in the future.

The first strategy was to switch which replica held the master. This would work in some cases because the master is the agent responsible for moving the Back Link obituaries through their various states. In the case where the replica was inconsistent and the master didn't hold the deleted object, switching masters to an agent that held the deleted entry with its obituaries would give the new agent the license to push the obituaries through their states and eventually purge it out. Send Single Entry is a much cleaner and less dangerous way to resolve obituaries that are stuck because the replica is inconsistent.

The second strategy used was to run DSRepair with certain switches to delete all obituaries. (There is a third-party application which resolves stuck obituaries by launching DSRepair.) We do not recommend this strategy. Using those switches will delete all obituaries on this agent, which means that obituaries that are not stuck might also be removed, creating further replica inconsistencies and more stuck obituaries. Because this is not a distributed operation, you must run DSRepair on all of the servers with stuck obituaries, which increases the odds that one of those servers has obituaries

for another partition which will be prematurely deleted. The premature deletion of obituaries can cause additional orphaned obituaries and, in turn, cause problems which can be found years later when you change replicas types, add new replicas, or perform other partitioning operations.

The third strategy used was to make objects authoritative, either using DSBrowse with the advanced mode operation and time stamping the entry, or running DSRepair with the -0T switch. This forces the entry to become authoritative and synchronize out to all other replicas. This should be done with great care because you might lose data changed on other servers. We recommend that this be a rarely employed method of obituary cleanup.

Migrating to OpenText eDirectory

Migrating the Sun ONE Schema to OpenText eDirectory

To migrate the Sun ONE schema to OpenText eDirectory, complete the following steps:

Step 1: Perform the Schema Cache Update Operation

You can write the errors encountered while comparing the schema to an error file using the following command:

```
ice -e LDIF error file name -C -a -SLDAP -s Sun ONE server -p Sun ONE port
-DLDAP -s eDirectory server -p eDirectory port
```

For example:

```
ice -e err.ldf -C -a -SLDAP -s sun_srv1 -p sun_port1 -DLDAP -s edir_srv2 -
p edir_port2
```

Any errors encountered while comparing the schema is written to the error file (`err.ldf` in the example). You do not need to login to perform this operation unless one of the servers require authentication in order to read the Root DSE. Microsoft Active Directory requires authentication to read the Root DSE.

Step 2: Rectify the Error LDIF File to Eliminate the Errors

- Sun ONE defines some schema definitions publicly that OpenText eDirectory does not. This includes attributes like `objectClasses`, `attributeTypes`, `ldapSyntaxes`, and `subschemSubentry`. These definitions exist internally and are very important to the schema, and therefore, they cannot be modified. Operations that try to modify these definitions results in the following error:

  ```
  LDAP error : 53 (DSA is unwilling to perform)
  ```

  Any records that contain references to these definitions cause the following error:

  ```
  LDAP error : 16 : ( No such attribute )
  ```

  Thus, records that contain any reference to these objects or that try to modify these definitions need to be commented in the LDIF error file (`err.ldf` in the example).

- Some objectClasses definitions in Sun ONE do not have naming attributes. Adding these objectClasses would result in the following error in OpenText eDirectory:

  ```
  LDAP error : 80 (NDS error: ambiguous naming (-651)
  ```

This error occurs because Sun ONE does not use the same method for determining naming rules as eDirectory.

To solve this, you can use any *one* of the three following options:

Option 1:

Go through each of the offending objectClasses and add a valid naming attribute to each of them.

For example:

To add the naming attribute `[ cn ]` to the objectClass `netscapeMachineData` modify the entry (that is *emphasized* in the example below) in the `err.ldf` file to include the `X-NDS_NAMING` flag as shown below:

```
dn: cn=schemachangetype: modifyadd: objectClassesobjectClasses: (
2.16.840.1.113730.3.2.32 NAME 'netscapeMachineData'
    DESC 'iPlanet defined objectclass' SUP top STRUCTURAL MAY c'n '
X-NDS_NAMING 'cn' )-
```

Option 2:

Go through each of the offending objectClasses and make them AUXILIARY or ABSTRACT.

For example:

To modify the definition of objectClass `netscapeMachineData` from STRUCTURAL to AUXILIARY, modify the `err.ldf` file entry (that is *emphasized* in the example below) as shown below:

```
dn: cn=schemachangetype: modifyadd: objectClassesobjectClasses: (
2.16.840.1.113730.3.2.32 NAME 'netscapeMachineData'
   DESC 'iPlanet defined objectclass' SUP top AUXILIARY )-
```

To modify the definition of objectClass `netscapeMachineData` from STRUCTURAL to ABSTRACT, modify the `err.ldf` file entry (that is *emphasized* in the example below) as shown below:

```
dn: cn=schemachangetype: modifyadd: objectClassesobjectClasses: (
2.16.840.1.113730.3.2.32 NAME 'netscapeMachineData'
   DESC 'iPlanet defined objectclass' SUP top ABSTRACT )-
```

Option 3:

Add `cn` to the definition of `Top` in OpenText eDirectory, which causes a potential naming attribute for all objectClasses.

There are two ways of adding `cn` to `Top`:

- **Method 1:**

  Create a file as shown below and name it `topsch.ldf`.

  ```
  version : 1

  dn:cn=schema

  changetype :modify

  delete : objectclasses

  objectclasses : ( 2.5.6.0 NAME 'top' STRUCTURAL )
  ```

```
-

add:objectclasses

objectclasses : (2.5.6.0 NAME 'top' STRUCTURAL MAY cn)
```

Use the following Import Conversion Export command line:

```
ice -SLDIF -f LDIF_file_name -DLDAP -s eDirectory_server -p
eDirectory_port -d eDirectory_Admin_DN -w eDirectory_password
```

For example:

```
ice -SLDIF -f topsch.ldf -DLDAP -s edir_srv2 -p edir_port2 -d
cn=admin,o=org -w pwd1
```

- **Method 2:**

  1. On the OpenText Identity Console home page, click **Schema Management** tile > **Attributes**.

  2. Click **Create Attribute** +.

  3. Provide **Attribute Name,** provide **Syntax**, and then **Add Attribute Flags**.

  4. Click **Create**.

- Some objectClass definitions contain `userPassword` as part of their mandatory attributes list. Adding such objectClasses to OpenText eDirectory cause the following error:

  ```
  LDAP error : 16 (No such attribute)
  ```

  To resolve this error, modify the objectClass definition to inherit the new objectClass from `ndsLoginProperties` and remove the `userPassword` attribute from the mandatory attribute list.

  For example:

  An objectClass containing `userPassword` in the mandatory attributes list:

  ```
  version : 1
  dn: cn=schemaz
  changetype: modify
  add: objectClasses
  objectClasses: ( 0.9.2342.19200300.100.4.19 NAME 'simpleSecurityObject'
  DESC '
   Standard LDAP objectClass' SUP top STRUCTURAL MUST userPassword )
  ```

  Needs to be modified as following (notice the change to the last line):

  ```
  version : 1
  dn: cn=schema
  changetype: modify
  add: objectClasses
  objectClasses: ( 0.9.2342.19200300.100.4.19 NAME 'simpleSecurityObject'
  DESC '
   Standard LDAP objectClass' SUP (ndsLoginProperties $ top) STRUCTURAL)
  ```

Step 3: Import the LDIF File

Use the following OpenText Import Conversion Export command to import the modified schema compare LDIF file (`err.ldf` in our example):

```
ice -e error_file -SLDIF -f modified_LDIF_file -DLDAP -s eDirectory_server
-p eDirectory_port -d eDirectory_Admin_DN -w eDirectory_password
```

For example:

```
ice -e errors.ldf -SLDIF -f err.ldf -DLDAP -s edir_srv2 -p edir_port2 -d
cn=admin,o=org -w pwd1
```

Migrating the Active Directory Schema to OpenText eDirectory Using ICE

While migrating schema from Active Directory to OpenText eDirectory using ICE, schema migration for the `Computer` objectClass fails with an `ambiguous naming error (-651)` error.

To resolve this, complete the following steps:

Step 1: Perform the Schema Cache Update Operation

While migrating schema from Active Directory to OpenText eDirectory using ICE, ensure that you have provided the error log option (`-e`) of ICE as follows:

```
ice -e error_file -S ldap -s Active_Directory_server -p
Active_Directory_port -d Active_Directory_full_admin_context -w
Active_Directory_password -D ldap -s eDirectory_server -p eDirectory_port -
d eDirectory_full_admin_context -w eDirectory_password
```

For example:

```
ice -e err.ldf -S ldap -s activesrv1 -p activeport1 -d cn=admin,o=company -
w activepwd -D ldap -s edirsrv2 -p edirport2 -d cn=admin,o=company -w
edirpwd
```

Step 2: Rectify the Error LDIF File to Eliminate the Errors

The failed entry would be present in the `err.ldf` file as shown below:

```
dn: cn=schema

changetype: modify

delete: objectclasses

objectclasses: ( 2.16.840.1.113719.1.1.6.1.4 NAME 'computer' )

-

add: objectclasses

objectclasses: ( 2.16.840.1.113719.1.1.6.1.4 NAME 'computer' SUP (device $
 user ) STRUCTURAL MAY (operator $ server $ status $ cn $ networkAddress $
 local PolicyFlags $ defaultLocalPolicyObject $ machineRole $ location $
 netbootInitialization $ netbootGUID $ netbootMachineFilePath $ siteGUID $
 operatingSystem $ operatingSystemVersion $ operatingSystemServicePack $
 operatingSystemHotfix $ volumeCount $ physicalLocationObject $ dNSHostName
 $ policyReplicationFlags $ managedBy $ rIDSetReferences $ catalogs $
 netbootSIFFile $ netboot MirrorDataFile ) X-NDS_NOT_CONTAINER '1' X
 -NDS_NONREMOVABLE '1' X-NDS_NAME 'Computer' )

-
```

Modify this entry in the error file (`err.ldf` in the example) to remove the `user` objectClass from the list of superior objectClasses in the definition of the `Computer` objectClass, as shown below:

```
dn: cn=schema

changetype: modify

delete: objectclasses

objectclasses: ( 2.16.840.1.113719.1.1.6.1.4 NAME 'computer' )

-

add: objectclasses

objectclasses: ( 2.16.840.1.113719.1.1.6.1.4 NAME 'computer' SUP device
 STRUCTURAL MAY (operator $ server $ status $ cn $ networkAddress $ local
 PolicyFlags $ defaultLocalPolicyObject $ machineRole $ location $
 netbootInitialization $ netbootGUID $ netbootMachineFilePath $ siteGUID $
 operatingSystem $ operatingSystemVersion $ operatingSystemServicePack $
 operatingSystemHotfix $ volumeCount $ physicalLocationObject $ dNSHostName
 $ policyReplicationFlags $ managedBy $ rIDSetReferences $ catalogs $
 netbootSIFFile $ netbootMirrorDataFile ) X-NDS_NOT_CONTAINER '1' X
 -NDS_NONREMOVABLE '1' X-NDS_NAME 'Computer' )
```

Step 3: Import the LDIF File

Now, import the modified entry using the following ICE command:

```
ice -S ldif -f LDIF_file -D ldap -s Novell_eDirectory_server -p port_number
-d full_admin_context -w password
```

For example:

```
ice -S ldif -f err.ldf -D ldap -s edirsrv1 -p edirport1 -d
cn=admin,o=company -w pwd1
```

Migrating from OpenLDAP to OpenText eDirectory

The data that is migrated from an OpenLDAP server can have MD5 passwords, which may cause the applications to break if the appropriate NMAS methods are not installed. The NMAS method, SimplePassword, needs to be installed for the OpenText eDirectory using the command as below:

```
nmasinst -addmethod admin_context treename configfile -h Hostname:port-w
password
```

For example: `nmasinst -addmethod admin.novell eDir-Tree /Linux/eDirectory/ nmas/NmasMethods/Novell/SimplePassword/config.txt -h eDir_srv:524 -w secret`

Migrating the OpenLDAP Schema to OpenTextv eDirectory

To migrate the OpenLDAP schema to OpenText eDirectory, complete the following steps:

Step 1: Perform the Schema Cache Update Operation

You can write the errors encountered while comparing the schema to an error file using the following command:

```
ice -e error_file -C -a -S ldap -s OpenLDAP_server -p Open_LDAP_port - D
ldap -s eDirectory_server -p eDirectory_port -d
eDirectory_full_admin_context -w eDirectory_password
```

For example:

```
ice -e err.ldf -C -a -SLDAP -s open_srv1 -p open_port1 -DLDAP -s edir_srv2
-p edir_port2 -d cn=admin,o=novell -w secret
```

Any errors encountered while comparing the schema is written to the error file (`err.ldf` in the example).

Step 2: Rectify the Error LDIF File to Eliminate the Errors

Open LDAP defines some schema definitions publicly, which include attributes like `objectClasses`, `attributeTypes`, `ldapSyntaxes`, and `subschemSubentry`. These definitions exist internally and are very important to the schema, and therefore, they cannot be modified. Operations that try to modify these definitions results in the following error:

```
LDAP error : 53 (DSA is unwilling to perform)
```

Any records that contain references to these definitions cause the following error:

```
LDAP error : 16 ( No such attribute )
```

Thus, records that contain any reference to these objects or that try to modify these definitions need to be commented in the LDIF error file (`err.ldf` in the example).

Migrating the Open LDAP Data to OpenText eDirectory

Execute the following command to migrate the data:

```
ice -e error_data.ldif -SLDAP -s OpenLDAP_server -p OpenLDAP_port  -d
admin_context  -w password -t -b dc=blr,dc=novell,dc=com -F objectclass=* -
DLDAP -d  admin_context -w password  -l  -F
```

For example:

```
ice -e err_data.ldif -SLDAP -s open_srv1 -p open_port1  -d
cn=administrator,dc=blr,dc=novell,dc=com -w secret1 -t -b
dc=blr,dc=novell,dc=com -F objectclass=* -DLDAP -d cn=admin,o=novell -w
secret2 -l -F
```

Some objects also may fail due to forward referencing and internal dependencies on the objects, which may not break any applications.

Making PAM Work with OpenText eDirectory After Migration

After migrating from OpenLDAP to OpenText eDirectory, you need to make some changes for PAM to work with OpenText eDirectory.

Changes in /etc/ldap.conf File

```
# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
  binddn cn=admin,o=acme
  ...
# The credentials to bind with.
# Optional: default is no credential.
  bindpw secret
  ...
# The search scope.
  scope sub
  ...
# Filter to AND with uid=%s
  pam_filter objectclass=inetorgperson
  ...
# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
  pam_password nds
  ...
  ssl off
  ...
```

Changes to the Data in the Directory

This change is only specific to the scenario where the users objects in OpenLDAP have CRYPT as the password hash algorithm.

Using OpenText Identity Console, add the following attribute with the specified value to the container having all the user objects:

Attribute: `sasDefaultLoginSequence`

Value: Simple Password

Troubleshooting Schema

This section includes information for troubleshooting schema.

Troubleshooting Schema

When an auxiliary class is disassociated with an object, the value is not immediately deleted, but it is marked as not present. The auxiliary class is associated with the entry until the DRL process cleans up these values during the actual object validation.

Because the DRL is a resource-consuming background process, other operations are slow during this cleanup. The duration of the cleanup process depends on the number of actual objects and external references in the system. Because this process is CPU and memory intensive, you should not run it frequently. By default, the Backlinker background process runs 50 minutes after ndsd is started and then subsequently runs every 13 hours.

Clearing an auxiliary class from an entry can take between 0 and 13 hours, plus the time taken to process this entry in the system.

To work around this issue, delete the entry of the auxiliary class by triggering the Backlinker through DSTrace or iMonitor.

NOTE: When the object is deleted, the values are immediately purged because this deletion is handled by other background processes.

Troubleshooting LDAP Syntax Mapping

Some of the DS syntaxes are not uniquely mapped to LDAP syntaxes. This occurs in OpenText eDirectory 9.1 and below.

This issue has been fixed with OpenText eDirectory 9.1 SP1. In case you want to go back to the previous mapping, set the `NDSD_NLDAP_PRE911_SCHEMA` environment variable to any value.

Troubleshooting DSRepair

Running DSRepair on an NFS Mounted DIB on Linux

You might get -732 or -6009 errors while trying to run the ndsrepair (DSRepair) operations on an NFS-mounted DIB on Linux systems.

Running DSRepair with -R Option Hangs

After enabling encrypted attributes on indexed attributes, if you run ndsrepair (DSRepair) with the -R option, it hangs.

Troubleshooting Replication

OpenText eDirectory offers the robust directory service and the fault tolerance inherent in replication. Replication allows you to keep copies of the OpenText eDirectory database, or portions of it, on multiple servers at once.

Configuring Encrypted Replication Through OpenText Identity Console

You cannot configure encrypted replication through OpenText Identity Console if any server in the replica ring is down.

Merging Trees With Encrypted Replication Fails

When encrypted replication is enabled, merging trees fails. Disable secure replication on each tree before doing a merge. If the trees have EBA enabled servers, then tree-merge may succeed. However, post-merge, the tree becomes unstable and the replication is improper causing authentications to fail

Recovering from OpenText eDirectory Replica Problems

You should always keep multiple replicas of OpenText eDirectory partitions. If you do so and one replica becomes corrupted or is lost because of a failed hard disk, you can delete that replica using OpenText Identity Console and replace it with a new one from the intact replica.

For more information on deleting replicas, see Administering Replicas in the *OpenText eDirectory 9.0 Administration Guide*.

Troubleshooting Clone DIB Issues

Clone DIB Fails With -601 and -603 Errors

When encrypted attributes and encrypted replication is enabled at the tree level, clone DIB fails with the following errors:

- Clone DIB on target server fails with the -601 error while configuring SAS
- After Clone DIB, the newly created clone object fails with the -603 error

To work around these issues, disable encrypted attributes and encrypted replication.

Clone DIB Can Fail Immediately After Offline Bulkload

If you try taking the clone of a server immediately after an offline bulkload, it might result in a failure, if the bulkload has been done with the disable indices option.

However, this is not an issue if the dibclone is initiated a few hours after the bulkload completion.

Issue in Cloning with Enabled Encrypted Replication Feature

While cloning with the Encrypted Replication feature enabled on the source server, modify the ER policy to temporarily exclude the cloned server. This can be changed after the configuration of the cloned server is complete.

Troubleshooting OpenText Public Key Infrastructure Services

PKI Operations Not Working

If PKI operations in OpenText Identity Console are not working, it could be because OpenText PKI Services are not running on the Linux. Start the PKI Services by entering `npki -l`.

If you cannot create certificates, you need to ensure that the NICI module has been properly installed. See Initializing the NICI Module on the Server. To verify if NICI is initialized, see Verifying Whether NICI Is Installed and Initialized on the Server.

Removing the configuration of an OpenText eDirectory server that is acting as a treekey server in a multiserver tree after having moved the existing OpenText eDirectory objects to a different server fails with the error code for Crucial Replica.

To complete the operation, change the Key Server DN attribute in the W0 object under Security Container > KAP to another server in the tree that has downloaded the treekey from this server.

1. On the OpenText Identity Console home page, click the **Object Management** tile.
2. On the Type drop down list > select **Security Domain Key List** > click **Search**.
3. Select the W0 object (usually W0.KAP.Security).
4. Select **NDSPKI:SD Key Server DN**, then click **Modify item** .

5 Specify the name and context of a different server in the **Security Domain Key Server's DN** field, then click **OK**.

6 Click **Apply**, then click **OK**.

While uninstalling the OpenText eDirectory Server holding the CA, the KMOs created on that server will be moved to another server in the tree and become invalid

You should re-create the CA and KMOs for the tree. See "Creating an Organizational Certificate Authority Object" and "Creating a Server Certificate Object" in the *OpenText™ eDirectory CE 24.4 (v9.3) Administration Guide* for more information.

We recommend that you do not uninstall the OpenText eDirectory server where the CA for the tree has been created.

Using PKIDiag

PKIDiag is a utility designed to diagnose and fix Certificate Server objects. PKIDiag can be used to do the following:

- Rename or move server-related objects so that they conform to the correct naming and containment scheme if a server has been moved.
- Create required objects if they do not exist.
- Grant the necessary rights between objects.
- Link objects if they are not linked.
- Create the SSL CertificateIP and the SSL CertificateDNS certificates if they do not exist.
- Fix the SSL CertificateIP and the SSL CertificateDNS certificates if either has an incorrect name, is out of date, or is close to expiring.

The PKIDiag functionality is used by two other processes, the server auto health check and the Create Default Certificate task in OpenText Identity Console.

The server auto health check is run whenever a server is restarted or whenever DSREPAIR is run. Create Default Certificate is a process you use to replace the default certificates created when you install Certificate Server. See "Creating Default Server Certificate Objects" on page 661 for more information.

See TID #3640106 for more information about PKIDiag and how it can be used.

Waiting for Servers to Synchronize

Occasionally, after the user certificate has been created, the client is unable to refresh the view to include the new certificate. A dialog box is shown with the message "Waiting for servers to synchronize." At this point, the user certificate has been created but the servers involved in the creation have not yet synchronized. You can close the dialog box without impacting the creation of the user's certificate.

Error Reusing Certificate Nicknames

If an error occurs during user certificate creation, try using a different nickname for the certificate. The nickname that was specified might not be available for reuse.

-1426 Error Exporting a User's Private Key

All servers with replicas of the partition in which the User object resides should have the same level of cryptography (U.S./Worldwide NICI or Import Restricted NICI). If they do not, an error of -1426 might appear when exporting the user's private key if the key size is too large.

To export the user's private key after a -1426 error has occurred, you must either upgrade the cryptography on the servers with replicas of the partition or remove the replica from those servers that have exportable cryptography.

Server Uses Expired SSL CertificateIP Certificate

OpenText eDirectory 9.0 does not support the SSL CertificateIP certificate. If you upgrade to OpenText eDirectory 9.0 from a previous version, the SSL CertificateIP certificate remains associated with the server. When the certificates in your environment expire, the SSL CertificateIP certificate does not renew automatically.

Any time after you upgrade to OpenText eDirectory 9.0, you can start using the SSL CertificateDNS certificate instead of the SSL CertificateIP certificate.

External CAs

Some third-party CAs such as VeriSign use an intermediate CA to sign server certificates. In order to import these certificates into a Server Certificate object, the server certificate as well as the Intermediate CA and the trusted root certificate must be in a single PKCS #7 formatted file (.P7B). If your CA cannot provide you with such a file, you can create one yourself by following these steps on a client machine with Internet Explorer 5.5 or later installed.

1. Import the server certificate into Internet Explorer. You can do this by double-clicking on the file or by selecting **File** > **Open** and selecting the filename.
2. If the external CA's certificate is not already listed as a trusted CA in Internet Explorer, import the Intermediate CAs as well as the root level CA in the same manner.
3. In Internet Explorer, select **Tools** > **Internet Options**. Select the **Content** tab, then select the **Certificates** button.
4. On the **Personal** tab, find the server certificate. Select it and click **Export**.
5. Accept the defaults in the wizard until you get to the Export File Format page, then select the Cryptographic Message Syntax Standard - PKCS #7 Certificates (.p7b) format.
6. Continue with the wizard.

   The PKCS #7 file can now be imported into the Server Certificate object.

Moving a Server

If a Server object is moved, the LDAP objects, SAS service object, and Server Certificate objects (Key Material Objects) for that server should also be moved. But the server auto health check will move the objects for you the next time you restart the server.

DNS Support

If DNS is configured for the server, the default subject name for a server certificate will be:

.CN=<*Server's DNS Name*>.O=<*Tree Name*>

Otherwise, the default subject name is the fully distinguished name of the server. You can modify the default subject name by selecting Custom during the certificate creation process.

Removing a Server from OpenText eDirectory

When removing a server from OpenText eDirectory™ and then reinstalling it into the same context with the same name, a successful re-installation occurs only if the SAS Service object representing the removed server is also deleted, if it existed.

The process should go like this:

1. Determine if the default certificates need to be backed up. If so, back them up.
2. Delete the default certificates.
3. Delete the SAS object.

For example, for a server named MYSERVER, a SAS object named SAS Service - MYSERVER could exist in the same container as the server. This SAS object must be manually deleted (using Identity Console) after the server is removed from the tree, but before the server is reinstalled into the tree.

If the server is the Organizational CA or the SD Key server, you must complete some additional steps. These steps are documented in TID #3623407.

The default server certificates created for the server should also be removed so that they are re-created when the server is reinserted.

These certificates are SSL Certificate IP - MYSERVER and SSL Certificate DNS - MYSERVER. You should be careful when deleting these certificates. If data has been encrypted by using either of these certificates, the data must be retrieved before the certificates are deleted.

Subject Name Limitations for CAs

Server certificates with an @ character in their subject names might cause SSL connections to fail. Contact Technical Support for a resolution of the problem.

Certificate Validation Speed

The certificate validation process includes several checks of the data in the certificate as well as the data in the certificate chain. A certificate chain is composed of a Root CA certificate and, optionally, the certificates of one or more intermediate CAs.

Validating the information in a certificate and its associated certificate chain is not a time-intensive process. However, there are occasions where the validation might take longer:

- If the certificate was signed by an external CA and one or more of the certificates has a CRL distribution point extension.

  In order to validate the certificate, the CRL for each applicable certificate in the chain must be retrieved. The CRL must then be examined to determine whether or not the certificate has been revoked.

If the CRLs are large or if the server operating the CRL distribution point is busy, it might take some time to validate a certificate. The time required can be decreased by doing one or more of the following:

- Upgrade the speed of the connection being used to check the revocation status of the certificate.
- Contact your CA provider.

- If one or more of the certificates has an OCSP AIA extension. If the OCSP responder is busy, it might take a significant amount of time to validate.
- If you are validating a user certificate.

  For server certificates, the entire certificate chain is stored along with the server certificate in the Key Material object. Therefore, when a server certificate is validated, the client can get all of the certificates necessary by simply reading one object. User certificates, however, are stored differently. Only the user certificate itself is stored in the User object. Thus, the client must retrieve the certificate chain from other objects stored in the Security container in order to validate the user certificate.

  In order to validate a user certificate signed by the Organizational CA, the client must read the Organizational CA's object in order to retrieve the CA's certificate. In order to validate a user certificate signed by an external CA, the client must read the Trusted Roots container in the Security container in order to compose a certificate chain that matches the user certificate. In the latter case, an Administrator must have already imported the certificates of the external CAs into the Trusted Roots container in order for the validation of the User certificate to succeed.

  The time required to validate a user certificate can be decreased by removing expired certificates that are no longer trusted from the Trusted Roots container.

Validating Certificates after Deleting the Organizational CA

If you delete the Organizational CA (other than during a backup and restore procedure), you should export the self-signed certificate and create a new trusted root in the trusted roots container. If you don't, you will experience the following behavior when validating these certificates:

- User certificates signed by the deleted CA are invalid. This is because the certificate of the CA that signed the user certificate could not be found in the Organizational CA object or in the Trusted Roots container. If you want those user certificates to remain valid, you must add the previous CA's self-signed certificate to the Trusted Roots container.
- Server certificates signed by the deleted CA continue to be valid. This is because the CA's certificate is stored in the Key Material object along with the server certificate.

  If you deleted the Organizational CA because the key had been compromised or because of some security breach, you should immediately revoke all user and server certificates that were signed by the CA. If you cannot revoke them, you should delete them and create new certificates to take their place. You should also tell all users who might have imported your Organizational CA's certificate into their browsers to delete the certificate.

Renaming the Security Container

You cannot rename the security container.

OpenText eDirectory Is Unable to Validate Certificates After Recreating the Tree CA

OpenText eDirectory is unable to validate certificates after recreating the CA when OpenText eDirectory is upgraded to the latest version or installed in a custom location.

To workaround this issue, if OpenText eDirectory installation path is anything other than `C:\NetIQ\eDirectory` (on Windows) and `/var/opt/novell/eDirectory` (on Linux), you must specify the correct CRL file path with respect to the OpenText eDirectory installation path when you recreate the TREE CA or while creating the CRL object. You must choose the Custom option in OpenText Identity Console > **Certificate Management** tile > **CA Management** while recreating the CA from the Configure Certificate Authority Wizard and specify the correct CRL file path to avoid any error. For example, OpenText eDirectory installs the CRL files in `C:\NetIQ\eDirectory\htdoc\crl\` path by default. In case you choose to install eDirectory in a custom location (`C:\`*`CustomLocation`*`\eDirectory\`), ensure to update the CRL file path while recreating the TREE CA. Such as `C:\`*`CustomLocation`*`\eDirectory\htdoc\crl\.`

NOTE: If OpenText eDirectory was installed in the default location (`C:\Novell\NDS\`) in any prior version, you still have to perform the workaround mentioned above while recreating the TREE CA after upgrading to the latest version.

OpenText eDirectory Displays an Error Message when User Tries to Change Password in Tree with DES Tree Key

After upgrading OpenText eDirectory server to 9.2 or above, if there is a DES tree key in the OpenText eDirectory tree, some users will not be able to change their passwords and error `-16060` is displayed. This happens due to passwords being protected with DES tree key (56-bit) and FIPS mode being enabled (`n4u.server.fips_tls=1`) in the `nds.conf` file.

To workaround this issue, perform the following steps:

1. Disable FIPS mode on your OpenText eDirectory server by setting `n4u.server.fips_tls` to `0`.
2. Generate a new 3DES tree key using SDIdiag 2.7.0 using the command `SD -G`.

   NOTE: To install SDIdiag 2.7.0, you must install eDirectory 8.8.8 P11 on a mew machine and then install SDIdiag 2.7.0 on the same machine.

   ---
3. Re-encrypt the passwords using the `Diagpwd` utility. For more information, see "Universal Password Diagnostic Utility" on page 741.
4. Enable FIPS mode on the OpenText eDirectory server by setting `n4u.server.fips_tls` to `1`.
5. Now you must be able to change user's password successfully. For more information, see TID 7024695.

OpenText eDirectory CA Fails to Create the Certificate Revocation List (CRL)

A new Certificate Authority (CA) in OpenText eDirectory fails to create the certificate revocation list (CRL). While trying to access the CRL from OpenText Identity Console, error -603 is displayed and the `ndspkiCertificateRevocationList` attribute is also found to be missing.

There is no workaround at this moment.

Troubleshooting Utilities on Linux

OpenText Import Convert Export Utility

If an LDAP server is refreshed or unloaded, while a OpenText Import Conversion Export operation is running, the LBURP `operation is timed out` message is displayed on the screen. The server recovers later, when the LBURP operation times out.

ndsmerge Utility

The PKI servers are not active after a merge operation. They must be restarted using the `npki -l` command.

Merge operations might not be successful on different versions of the product. If your server is running an older version of NDS or OpenText eDirectory, update to the latest version of OpenText eDirectory, then continue the merge operations.

The merging of two trees will not succeed if containers with the same name subordinate to a tree are present in both the source and target trees. Rename one of the containers, then continue with the merge operation.

During the graft operation, error message -`611 Illegal Containment` might appear. Modify the schema by running `ndsrepair`. Then run `ndsrepair -S` and select **Optional Schema Enhancements**.

If you try to perform an incremental backup before performing a full backup, the backup operation will fail.

DSTrace Utility

When you turn on the DSTrace screen, an error message might display indicating that a primary object is invalid for the reference link. You can ignore this message if OpenText eDirectory is functioning correctly.

ndsbackup Utility

While backing up OpenText eDirectory, `NDS Error: Connect to NDS server failed` might display. This might be caused by OpenText eDirectory listening on a port other than the default port 524. At the command line, enter the port number that OpenText eDirectory was configured on. For example, if OpenText eDirectory is configured on port number 1524, enter the following:

```
ndsbackup sR 164.99.148.82:1524
```

While you back up the data, OpenText eDirectory might display `NDS Error: Requires a Password` error. This is because the server might have attributes marked for encryption and you might not have used the option -E to encrypt or decrypt the backup data.

If you try to perform an incremental backup before performing a full backup, the backup operation will fail.

Error -786 While Running DSRepair

When using DSRepair, you need to have three times the size of DIB free in the specific partition of your machine in which DSRepair is running.

OpenText eDirectory Utilities Require Users to Authenticate Using NDS password

If Universal Password is being used, then it must be synced to the NDS `password` in order for all OpenText eDirectory command line tools to authenticate.

Troubleshooting NMAS

NMAS Error Codes

A complete list of NMAS error codes can be found in the NMAS NDK.

Login Method and Sequence Issues

- For products to use NMAS login methods properly, at least one NMAS server in the OpenText eDirectory partition needs to hold a R/W replica of the User objects that will be using NMAS.
- Not all login or post-login methods use the initial password field when they are activated. If you are prompted to enter a password, you can ignore the password field and close it.
- Two password methods, such as Simple and NDS, cannot be used in an AND sequence if the Novell Client is set to display the password field, which is the default.

Administration Issues

- You must give explicit rights to users with graded authentication. Inherited rights do not work. For example, an administrator's Supervisor right is defined at the [Root] container. Rights for the administrator are not defined in the Volume object. If the administrator changes the volume's security label from Logged In to any other security label, the administrator cannot get the appropriate rights. The administrator must assign explicit rights to the volume, directories, or files in the volume.
- If Universal Password is enabled and you attempt to set the simple password, a -1697 error message is returned.
- OpenText eDirectory utilities like DSBackup (ndsbackup), DSRepair (ndsrepair), and DSMerge (ndsmerge) work with NDS passwords alone but do not work with NMAS Simple password. OpenText eDirectory 9.0 uses Universal Password.

  For information on Universal Password, see the *Password Management 3.3.2 Administration Guide* (https://www.netiq.com/documentation/password_management33/pwm_administration/data/bookinfo.html).
- Clicking **OK** or switching between tabs when creating or renaming a label always creates or renames the label even if you respond **No** to the **Save Changes made for Labels?** prompt. You must click the **Cancel** button to cancel any changes. After a label is created, it cannot be deleted. However, you can rename it to an unused name, such as `Unused_x`.
- When you use XDAS auditing for NMAS, the DN format of the following events is not generated in the LDAP notation.
  - 00290035 SASL Mechanism Result
  - 00290061 Set Login Configuration
  - 00290062 Get Login Configuration
  - 00290064 Set Login Secret

NOTE: The ID (for example, 00290035 or 00290061) specifies the NMAS event ID as mentioned in the `lsc` file. The NMAS event ID is part of the `subEvent` field in the XDAS format.

Unable to Log In Using Any Method on Linux

After installing and configuring NMAS, restart the OpenText eDirectory server.

After reinstalling a method after you have uninstalled a previous instance of that method, restart the OpenText eDirectory server.

The User Added Using the ICE Utility Is Unable to Log In Using Simple Password on Linux

While adding users with simple passwords through the OpenText Import Conversion Export utility, use the `-l` option.

SLP_NETWORK_ERROR(-23) Occurs in Windows Machines

The Service Location Protocol (SLP) query returns -23 SLP_NETWORK_ERROR on a virtual machine having a DHCP address or on a physical or a virtual machine in which SLP is not broadcasted.

You can avoid the SLP error by configuring the Directory Agent in your network in one of these ways:

1. Copy the `C:\Windows\System32\Novell\eDir\OpenSLP\slp.conf` file to the `c:\Windows\` directory.
2. Open the `slp.conf` file by using a text editor and change the following line:

   ```
   ;net.slp.DAAddresses = myDay1,myDa2,myDa3
   ```

   to

   ```
   net.slp.DAAddresses = <Give your DA Address>
   ```

3. Save the changes, then close the file.

OR

1. Copy the `C:\Windows\System32\Novell\eDir\OpenSLP\slp.conf` file to the `c:\Windows\` directory.
2. Open the `slp.conf` file by using a text editor and change the following line:

   ```
   ;net.slp.isDA = true
   ```

   to

   ```
   net.slp.isDA = true
   ```

3. Save the changes, then close the file.

Incorrect Installation Path Appears in the Installation Path Field During OpenText eDirectory Installation on Windows

While installing OpenText eDirectory, instead of accepting the default location for installing, if you click the **Browse** icon to select another location, and then close the Browse dialog without selecting any folder, incorrect installation path is displayed in the **Installation Path** field. This issue is found while installing OpenText eDirectory on Windows Server 2012 Standard Edition (64-bit) and Windows Server 2012 R2 (64-bit) only.

To workaround this issue, manually change the path to the desired location.

Adding a Server Fails if SLP is Not Configured Properly on Windows

Installing OpenText eDirectory fails while adding a server to a tree (where you have to browse your current tree), if SLPD is already installed and running. Windows displays a message, *launch.exe died*.

To successfully install OpenText eDirectory, perform the following steps without rebooting the system:

1. Stop Service Location Protocol Service.
2. Delete the `C:\Windows\slp.conf` file.
3. Delete the `C:\Windows\System32\Novell\eDir\OpenSLP` folder.
4. Delete the RegKeys for the SLPD service from `Registry HKLM\SYSTEM\CurrentControlSet\Services\slpd`.
5. Run the setup again with the Administrator role.

Accessing HTTPSTK When Directory Service Is Not Loaded

You can set up a preconfigured admin user that allows access to the HTTP Protocol Stack (HTTPSTK) when DS is not loaded. The preconfigured admin user, sadmin, has rights that are equivalent to the OpenText eDirectory Admin User object. If the server is in a state where eDirectory is not functioning correctly, you can log in to the server as this user and perform all the diagnostic and debugging tasks necessary that do not require OpenText eDirectory.

Setting the sadmin Password on Windows

Use the DHost remote manager page (accessible through the `/dhost` URL or from the root page) to set the sadmin password. `dhost.exe` must be running on the OpenText eDirectory server in order for you to set or change the sadmin password.

1. Open a Web browser.
2. In the address (URL) field, enter the following:

   `http://`*`server.name`*`:`*`port`*`/dhost`

   for example:

   `http://MyServer:80/dhost`

   You can also use the server IP address to access the DHost iConsole. For example:

`http://137.65.135.150:80/dhost`

3. Specify a username, context, and password.
4. Click **HTTP Server**, then specify an sadmin password.
5. Verify the password you just specified, then click **Submit**.

Setting the sadmin Password on Linux

To set the sadmin password on Linux, you can use either the DHost Remote Management page or the ndsconfig utility.

Using the DHost Remote Management Page

Access the DHost Remote Manager page through the `/dhost` URL or from the root page and set the sadmin password. OpenText eDirectory server must be running in order for you to set or change the sadmin password.

1. Open a Web browser.
2. In the address (URL) field, enter the following:

   `http://`*`server.name`*`:`*`port`*`/dhost`

   for example:

   `http://MyServer:80/dhost`

   You can also use the server IP address to access the DHost iConsole. For example:

   `http://137.65.135.150:80/dhost`
3. Specify a username, context, and password.
4. Click **HTTP Server**, then specify an sadmin password.
5. Verify the password you just specified, then click **Submit**.

Using the ndsconfig

Use the ndsconfig utility to set the sadmin password. ndsd must be running on the OpenText eDirectory server in order for you to set or change the sadmin password.

Enter the following at the server console:

`ndsconfig set http.server.sadmin-pwd=`*`password`*

where *password* is the new sadmin password.

For more information on using the ndsconfig utility, see “ndsconfig Utility Parameters” in the *OpenText™ eDirectory CE 24.4 (v9.3) Installation Guide*.

Troubleshooting Data Encryption

In OpenText eDirectory 9.0, you can encrypt specific sensitive data while they are stored on the disk and while they are accessed by the client. This chapter provides you information on the errors you might encounter while using the encrypted attributes and replication features in OpenText eDirectory 9.0.

For information on other error messages in OpenText eDirectory, refer to the OpenText Error Codes Website (http://www.novell.com/documentation/nwec/)

-6090 0xFFFFE836 ERR_ER_DISABLED

The OpenText eDirectory replica synchronization process tried to start encrypted replication with the target server. But the target OpenText eDirectory server has the encrypted replica synchronization process disabled

Possible Cause

Encrypted replication is disabled on the target OpenText eDirectory server.

Action

Enable encrypted replication on the target OpenText eDirectory server.

-6089 0xFFFFE837 ERR_REQUIRE_SECURE_ACCESS

An application (client access) tried to access an encrypted attribute over a clear text channel.

Source

OpenText eDirectory or NDS.

Possible Cause

The encrypted attributes are configured to be accessed only over a secure channel. The application is trying to access the encrypted attributes over a clear text channel.

Action

The application should access the encrypted attributes through a secure channel, like LDAP secure channel or HTTP secure channel.

Possible Cause

If you get this error during replication, one or more servers in the replica ring have some attributes marked for encryption and are configured to be accessed only over secure channel.

Action

Change the configuration of the encrypted attribute policy, so that the encrypted attributes can be accessed over insecure channels. For more information, see Chapter 11, “Encrypting Data in OpenText eDirectory,” on page 293.

Possible Cause

If you get this error when encrypted replication is configured at the partition level or between the replicas of the partition, then the replica ring has pre-eDirectory 8.8.x servers in it.

Action

Upgrade all the servers in the replica ring to a version compatible with eDirectory 8.8.x.

-666 FFFFFD66 INCOMPATIBLE NDS VERSION

Text goes here

Possible Cause

If encrypted replication is enabled at a partition level and if you are trying to add a replica of this partition to an OpenText eDirectory server, then the OpenText eDirectory version on this server is incompatible with the version on the source server.

Action

Upgrade the server to a compatible version of OpenText eDirectory.

Possible Cause

If the parent partition has pre-eDirectory 8.8.x servers (mixed version ring) and if the child partition has ER enabled, the merge and/or join partition operations would be disallowed and the ERR_INCOMPATIBLE_DS_VERSION error will be returned.

The reason for this is that the child partition contains sensitive data with ER enabled at the partition level and the parent partition having pre-eDirectory 8.8.x server. With ER enabled only between eDirectory 8.8.x servers, on merging, sensitive data is exposed when replicating to pre-eDirectory 8.8.x servers.

Action

1. Upgrade the server to a compatible version of OpenText eDirectory.

   OR

2. Disable ER at the parent or child partition.

   NOTE: On disabling ER, replication will happen in the clear text form.

Problem With Duplicate Encryption Algorithms

If you add an attribute for encryption using LDIF, do not associate duplicate algorithms with one attribute.

For example, marking *title* as an encrypted attribute with AES and DES encryption algorithms makes it unclear as to which algorithm is ultimately considered. Each time when limber is run it appears the title attribute toggles between AES and DES. Therefore, it seems as though there were some configuration changes.

To prevent such scenarios, we recommend you to avoid duplicate algorithms been assigned to the same attribute.

Encryption of Stream Attributes

Stream attributes might be present as clear text data. This is because OpenText eDirectory 9.0 does not encrypt stream attributes.

Configuring Encrypted Replication through OpenText Identity Console

You cannot configure encrypted replication through OpenText Identity Console if any server in the replica ring is down.

Viewing or Modifying Encrypted Attributes through OpenText Identity Console

If an attribute of an object is encrypted, you cannot view or modify the object by using OpenText Identity Console.

To work around this issue, you can view or modify the encrypted attribute over a secure channel, using any of the following methods:

- LDAP: The LDAP request must be send over a secure channel, which means that the trusted root certificate of the server must be used.
- ICE: LDIF scripts can be used to modify the object. If you do this, ICE must use a secure channel.
- Use OpenText Identity Console 1.7, or later.

NOTE: We recommend using OpenText Identity Console 1.7 or later for viewing or modifying encrypted attributes.

Alternatively, you can turn off the secure channel required option for viewing or modifying the encrypted attributes by disabling the `requireSecure` attribute in the EA policy. This makes the object and the encrypted attributes accessible by any client over clear text channel. After this, OpenText Identity Console will be able to access the object.

Merging Trees With Encrypted Replication Enabled Fails

When encrypted replication is enabled, merging trees fails. Disable secure replication on each tree before doing a merge.

Limber Displays -603 Error

Limber displays the -603 error if the server has only sub-ref replica of the encrypted attribute policy partition.

To work around this issue, do any one of the following:

- Give read access to the NCP server object. You can do this through OpenText Identity Console by adding a trustee at the tree root and giving read access to NCP server object. In the attributes, specify `attrEncryptionDefinition` and `attrEncryptionRequiresSecure`.
- Give Public Read access to the following attributes through LDAP or ndssch:
  - `attrEncryptionDefinition`
  - `attrEncryptionRequiresSecure`

The OpenText eDirectory Management Toolbox

The OpenText eDirectory Management Toolbox (eMBox) lets you access all of the OpenText eDirectory backend utilities remotely as well as on the server.

eMBox works with OpenText Identity Console to provide Web-based access to OpenText eDirectory utilities such as DSRepair, DSMerge, Backup and Restore, and Service Manager.

IMPORTANT: Roles and Access Control must be configured through OpenText Identity Console to the tree that is to be administered in order for eMBox tasks to be run.

All functions are accessible, either on the local server or remotely, through a command line client. You can perform tasks for multiple servers from one server or workstation using the eMBox Client. To run all OpenText eDirectory Management Tools (eMTools), including Backup, DSRepair, DSMerge, Schema Operations, and OpenText eDirectory Service Manager, eMBox must be loaded and running on the OpenText eDirectory server.

Unable to Stop the eMTool Services

When running the command `serviceStop -n{service}`, where *{service}* is one of the services (`libsasl.so`, `libncpengine.so`, `libhttpstk.so`, or `libdsloader.so`), the following error occurs:

`Service {service} could not be stopped, Error : -660`

This is not an error. You cannot stop these processes (specifically `libsasl.so`, `libncpengine.so`, `libhttpstk.so`, and `libdsloader.so`), because there are other modules dependent on them.

Restore gives -6020 error

If you have roll forward logs in a default location, while performing Restore operation using DSBK or eMBox Client, you will get -6020 error. To avoid this error, you need to give `-s` switch in the `restore` command.

Deletion of a Moved Object

Deletion of a moved object might fail (error -637) in a tree with two or more servers.

Issue with Moving a Dynamic Group

Moving a Dynamic Group object with `dynamicgroup` in the `Object Class` attribute to another container breaks the dynamic group functionality. After the move, queries and searches on dynamic members do not work.

Issue with Repairing Network Addresses through

When you repair the network addresses through eMBox, it throws the following errors because eMBox is not updated with the recent fixes for repair:

`ERROR: Could not find a net address for this server - Error : 11004`

`ERROR: Could not connect. Error : 11004`

Viewing French Man Pages

To view the French man page on Red Hat Linux, export the following:

```
export MANPATH=/opt/novell/man/frutf8:/opt/novell/eDirectory/man/frutf8
```

Deleting a Moved Object

Deletion of a moved object might fail (error -637) in a tree with two or more servers.

OpenTexte Directory Does Not Generate a Logout Event due to OpenText eDirectory Client Limitation

OpenText eDirectory does not generate a Logout event when you log out of OpenText Identity Console. This is because of a technical limitation in the client part of OpenText eDirectory.

Auditing applications can use NWDS APIs to receive logout events. Applications that use LDAP can monitor logout with unbind events.

Issues Generated by TERM While Running DSTrace

TIME and TAGS tags are displayed as enabled (underlined), but not by default. When the TERM is set to VT100 or xterm from a Linux terminal, these tags are displayed as if they are enabled (underlined). This issue does not occur for any other term, such as dtterm.

eMBox Does Not Handle Double-Byte Characters

eMBox does not handle double-byte characters for setting a roll-forward directory through the eMBox client and OpenText Identity Console. This can still be done by using DSBK.

Troubleshooting Issues with SASL-GSSAPI

This section discusses the error messages logged by the SASL-GSSAPI authentication mechanism.

Issue with Multiple User Objects

LDAP bind with SASL GSSAPI fails if the same Kerberos principal is associated with multiple OpenText eDirectory user objects.

Authorization ID

RFC2222 specifies support for an authorization ID sent by the user and client. This is not supported by the SASL GSSAPI method.

Log File

Error messages are logged in the `ndsd.log` file in Linux installations.

Error Messages

Error Message	Cause
SASL-GSSAPI: Reading Object user_FDN FAILED OpenText eDirectory error code	This error is generated in OpenText eDirectory. The Kerberos principal name is not attached to the user object (`userdn`).
SASL-GSSAPI: Reading Object Realm_FDN FAILED OpenText eDirectory error code	This error is generated in OpenText eDirectory. The `realm` object does not exist.
SASL-GSSAPI: Not enough memory	Not enough memory to perform the specific operation.
SASL-GSSAPI: Invalid Input	Input from client is defective or invalid
SASL-GSSAPI: NMAS error NMAS error code	This error is generated in NMAS and is an internal error.
SASL-GSS: Invalid LDAP service principal name *LDAP_service_principal_name*	The LDAP service principal name is invalid.
SASL-GSS: Reading LDAP service principal key from OpenText eDirectory failed	**Cause:** The LDAP service principal object is not created.
	Cause: The realm object's master key is changed.
	Cause: The LDAP service principal object was not found in the subtree of the realm to which it belongs.
SASL-GSS: Creating GSS context failed	**Cause:** The time is not in sync between the client, KDC and the OpenText eDirectory servers.
	Cause: The key of the LDAP service principal was changed in the Kerberos database, but not updated in OpenText eDirectory.
	Cause: The encryption type is not supported.
SASL GSSAPI: Invalid user FDN = *user_FDN*	The user FDN provided by the client is not valid.
SASL GSSAPI: No user DN is associated with principal *client_principal_name*	A `user` object under the subtree is not attached with the Kerberos principal name.
SASL GSSAPI: More than one user DN is associated with principal *client_principal_name*	More than one `user` object under the subtree is associated with the same principal.

Error Message	Cause
ldap_simple_bind_s: Invalid credentials major = 1, minor =0	**Cause:** The cause might be the version mismatch between the LDAP service principal on the KDC server and the LDAP service principal on the OpenText eDirectory server. This is because every time you extract the LDAP service principal key to the keytab file, the key version number gets incremented. **Action:** Complete the following procedure: 1. Update the key in OpenText eDirectory server so that the version numbers are in sync. 2. Destroy the tickets at the client. 3. Get the TGT again for the principal. 4. Perform the LDAP `sasl` bind operation.

Managing Error Logging in OpenText eDirectory

Error logging is automatically started during OpenText eDirectory installation.

Message Severity Levels

All the messages have a severity level attached to it that helps you determine how critical the message is.

Error Message	Description
Fatal: A fatal message indicates a significant problem, such as loss of data or functionality.	**Examples:** ◆ If the OpenText eDirectory server fails to load system modules like NCPEngine and DSLoader while loading modules, a fatal error is reported and logged. ◆ If the OpenText eDirectory server fails to bind on secure port 636, then a fatal error is reported and logged.

Error Message	Description
Warning: A message that is not necessarily severe, but might be a possible cause for future problem.	**Examples:** ◆ Connection failures between any two servers in tree, resulting in server getting added to bad address cache. Server can recover from this particular state by resetting the bad address cache. ◆ If the LDAP client application does a bind and closes the connection without doing an unbind then LDAP server should log warning with appropriate warning message. ◆ If the OpenText eDirectory server has consumed all the file descriptors and it has reached the Threshold limit as result server is not able to process any incoming requests and respond it and leading to failure of the application.
Error: A message that could be due to invalid operation, but which will not cause any problem.	**Examples:** ◆ When an client application tries to add a object for which attributes definition are not defined In schema, then OpenText eDirectory server will report the ERR_NO_SUCH_ATTRIBUTE error. ◆ When an User tries to login with an invalid password, OpenText eDirectory server will report error ERR_FAILED_AUTHENTICATION.
Information: A message that describes successful completion of an operation or event in the OpenText eDirectory server.	**Examples:** ◆ When a module gets loaded/unloaded successfully, it may be appropriate to log an informative message of the operation. ◆ When database cache configuration is changed, informative message should be logged on successfully saving the configuration.
Debug: A message that contains information which will help developers in debugging a program.	**Examples:** While doing a dynamic group search, display all the dynamic group members with information of entry ID, partition ID, and DN of the members. This information will help in knowing that all members are returned at the OpenText eDirectory level.

Configuring Error Logging

Setting the Severity Level on Linux: To configure the error logging settings for the server-side messages, you can use the `n4u.server.log-levels` and `n4u.server.log-file` parameters in the `/etc/opt/novell/eDirectory/conf/nds.conf` configuration file.

The severity levels available are `LogFatal`, `LogWarn`, `LogErr`, `LogInfo`, and `LogDbg` levels (in decreasing order of severity). For more information on the severity levels, see “Message Severity Levels” on page 845.

By default, the severity level is set LogFatal. So, only messages with severity level fatal will be logged.

To set the severity level, use the n4u.server.log-levels parameter in the nds.conf file as follows:

n4u.server.log-levels=*severity_level*

For example:

- To set the severity level to LogInfo and above, type the following:

  n4u.server.log-levels=LogInfo

  With this configuration, messages with severity level LogInfo and above (that is, LogFatal, LogWarn, and LogErr) will be logged into the log file.
- To set the severity level to LogWarn and above, type the following:

  n4u.server.log-levels=LogWarn

  With this configuration, messages with severity level LogWarn and above (LogFatal) will be logged into the log file.
- To set the severity level to LogDbg and above, type the following:

  n4u.server.log-levels=LogDbg

  With this configuration, messages with severity level LogDbg will be logged into the log file.

  NOTE: You must set the environment variable NDSD_EVENT_DISK_CACHE to true while setting the log level (n4u.server.log-levels) to LogDbg.

Specifying the Log File Name on Linux: To specify the location of the log file where the messages will be logged, use the n4u.server.log-file parameter in the nds.conf file. By default, the messages are logged into the ndsd.log file.

For example, to log the messages to /tmp/edir.log, type in the following:

n4u.server.log-file=/tmp/edir.log

To log the messages in the system log, use the n4u.server.log-file parameter as follows:

n4u.server.log-file=syslog

Setting the Severity Level on Windows

The severity levels available are LogFatal, LogWarn, LogErr, LogInfo, and LogDbg levels (in decreasing order of severity). For more information on the severity levels, refer to "Message Severity Levels" on page 845.

To set the severity level, do the following:

1. Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**
2. In the **Services** tab, select **dhlog.dlm**.
3. Enter the log level in the **Startup Parameters** box.

   For example, to set the log level to LogErr and above, enter the following:

`LogLevels=LogErr`

4 Click **Configure**

5 In the **ACS Config** tab, click the plus sign of **DHostLogger**.

The `LogLevel` parameter is updated with the configured value.

NOTE: Trace level severity does not work on Windows.

Specifying the Log File Name and Path on Windows

1 Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**

2 In the **Services** tab, select **dhlog.dlm**.

3 Enter the log file path in **Startup Parameters** as follows:

`LogFile=`*`file_path`*

For example, to set the log file path to `/tmp/Err.log`, enter the following in startup parameters:

`LogFile=/tmp/Err.log`

4 Click **Configure**

5 In the **ACS Config** tab, click the plus sign of **DHostLogger**.

The `LogFile` parameter is updated with the configured value.

Specifying the Log File Size on Windows

1 Click **Start** > **Settings** > **Control Panel** > **OpenText eDirectory Services**

2 In the **Services** tab, select **dhlog.dlm**.

3 Enter the log file path in **Startup Parameters** as follows:

`LogSize=`*`size`*

The default file size is 1 MB.

4 Click **Configure**

5 In the **ACS Config** tab, click the plus sign of **DHostLogger**.

The `LogSize` parameter is updated with the configured value.

Miscellaneous

Backing Up a Container

While using ndsbackup to backup a container that has many objects (like a million), it might take some time to get the list of the objects in the container and start their individual backup.

Repeated OpenText eDirectory Logins

Repeated OpenText eDirectory logins can use up the available memory. Disable the Login Update attribute using iMonitor to overcome this problem.

Enabling Event System Statistics

Time related statistics are maintained for every event thrown and consumed in OpenText eDirectory. This information is useful for troubleshooting event consumer issues. These statistics are not required for normal functioning of directory; therefore, they are disabled for performance reasons. Event statistics can be enabled at runtime by using iMonitor advanced configuration parameters.

To view the event statistics, set the ENABLE_EVENT_STATISTICS parameter and restart the server. It is a permanent configuration parameter.

Tracking Memory Corruption Issues on Linux

On Linux platforms, OpenText eDirectory uses Google malloc (`libtcmalloc`) as the default memory allocator.

To track memory corruption issues, set the `MALLOC_CHECK_ environment variable` in the ndsd startup script. The startup script checks for this variable. If set, the default system malloc is used, else `libtcmalloc` is loaded.

MALLOC_CHECK Settings in ndsd

- When `MALLOC_CHECK_` is set to 0, any detected heap corruption is silently ignored.
- When `MALLOC_CHECK_` is set to 2, abort is called immediately.

  This helps to identify the real cause of the memory corruption at early stages, which might be difficult to track later.

TCP Connection not Terminating after Abnormal Logout

Sometimes the OES Linux server fails to detect a client host that has gone down abruptly due to a workstation crashing or a power outage. However, the connection is active for the default timeout (about 12 to 15 minutes) before the connection is cleared. If you have set the concurrent connections to 1, it is recommended that you either terminate the connection manually, or wait for the estimated timeout before logging in again. This situation occurs when the watchdog process fails to close the connection cleanly. So, if the concurrent connections are set to 1 and the connection is not cleared by the watchdog, users cannot log in. Linux kernel provides three parameters to change the way `keepalive` probes work from the server side. Use these parameters to implement a workaround at the TCP level.

These parameters are available in `/proc/sys/net/ipv4/` directory.

- `tcp_keepalive_time`: Determines the frequency of sending the TCP `keepalive` packets to keep a connection alive if it is currently unused. This value is used only when `keepalive` is enabled.

  The `tcp_keepalive_time` takes an integer value in seconds. The default value is 7200 seconds or 2 hours. This holds good for most of the hosts and does not require many network resources. If you set this value to low, it engages your network resources with unnecessary traffic.
- `tcp_keepalive_probes`: Determines the frequency of sending TCP `keepalive` probes before deciding a broken connection.

  The `tcp_keepalive_probes` takes an integer value, recommended less than 50 depending on your `tcp_keepalive_time` and the `tcp_keepalive_interval` values. The default is to set to 9 probes before informing the application of the broken connection.

- `tcp_keepalive_intvl`: Determines the duration for a reply for each `keepalive` probe. This value is important to calculate the time before your connection has a `keepalive` death.

  The `tcp_keepalive_intvl` takes an integer value, the default is 75 seconds. So, 9 probes with 75 seconds each will take approximately 11 minutes. The default values of the `tcp_keepalive_probes` and `tcp_keepalive_intvl` variables can be used to evaluate the default time before the connection is timed out because of `keepalive`.

Modify these three parameters in a way that the change does not generate a lot of extra network traffic and still solves the problem. A sample modification could be as follows (a 3-minute detection time):

- `tcp_keepalive_time set -120`
- `tcp_keepalive_probes - 3`
- `tcp_keepalive_intvl - 20`

NOTE: Be careful with the parameter settings and avoid setting the already valid connections.

The settings take effect immediately after the files are modified. You need not restart any services. However, the settings are valid for the current session only. Once the server is re-booted, the settings revert to the default settings.

To make the setting permanent (even after a reboot), do the following:

Add the following entries in `/etc/sysctl.conf`.

- net.ipv4.tcp_keepalive_time=120
- net.ipv4.tcp_keepalive_probes=3
- net.ipv4.tcp_keepalive_intvl=20

We recommend these settings only if all the clients and servers are connected through LAN.

NDS Error, System Failure (-632) Occurs When Doing ldapsearch for the User Objects

Import the user objects with simple password and then enable universal password for the container where the user objects are imported. Stop the DS server and set the environment as `NDSD_TRY_NMASLOGIN_FIRST=true` and then start DS Server. When you do an ldapsearch for the user objects, which were imported with simple password, you get the following error:

```
ldap_bind: Unknown error,  additional info: NDS error: system failure (-
632)
```

To resolve this issue, set the default login sequence as simple password for the container where user objects are imported before doing ldapsearch for those user objects.

When LDAP requests NMAS to log in a user, NMAS uses the default login sequence. If you do not specify a default login sequence for these users, then it will use the NDS sequence. If these users are not given an NDS password when you imported them, then the NDS sequence will not work. If you enable universal password, then the simple password will be synchronized with the NDS password and universal password when the user logs in with the simple password.

Disabling SecretStore on Linux

An OpenText eDirectory administrator can disable SecretStore on Linux using the following processes:

1. Go to the `nds-modules` directory and rename or move the following SecretStore modules:

   ```
   libsss.so
   libssncp.so
   libssldp.so
   ```

2. Restart the server.

Disabling SecretStore on Windows

An OpenText eDirectory administrator can disable SecretStore on Windows using the following processes:

1. Go to the `novell\nds` directory and rename or move the following SecretStore modules:

   ```
   lsss.dll
   sss.dlm
   ssncp.dlm
   ssldp.dlm
   ```

2. Restart the server.

dsbk Configuration File Location

The `dsbk.conf` file is located in `/etc` instead of the location relative to the specific instance of OpenText eDirectory.

ldif2dib Fails to Open the Error Log File When the DIB Directory Exists In the Custom Path

ldif2dib fails to open the default log file, `ldif2dib.log` when the `dib` directory is relocated to a custom location.

To work around this issue, explicitly provide the log file location by using the `-b` switch.

ndsd Does Not Start After a System Crash

In some situations, OpenText eDirectory services (`ndsd`) doesn't start after a system crash or a power failure. To start the OpenText eDirectory again, do the following:

1. Delete `/var/opt/novell/eDirectory/data/ndsd.pid` file.
2. Enter `/etc/init.d/ndsd start` command.

Do not Execute DSTrace With All Tags Enabled on Linux Computers

With all tags enabled, ensure you do not run DSTrace on the following:

- **A loaded system in Journal mode:** It tends to build up ndsd memory.
- **Servers in inline mode:** It crashes ndsd.

LDAP is Not RFC Compliant For Anonymous Search Requests

If a client performs an unauthenticated search operation when anonymous binds are disabled, the LDAP server responds with the bind result of inappropriate authentication instead of the search result, `operationsError`.

Troubleshooting Ports with Custom OpenText eDirectory 9.0 Instances

In OpenText eDirectory 9.0, if you configure a new instance in a custom location when the default instance server is down, it takes the default instance ports. The default instance does not come up, because the ports of the default instance are allotted to the custom location instance.

Follow the procedure in "Troubleshooting Ports with Custom eDirectory 8.8 Instances" before rebooting the host.

Rebooting the Host

Only the default instance created through using the default instance binaries is started after reboot.

You can set the paths and use ndsmanage to start the other instances.

ndsd Not Listening at the Loopback Address on a Given NCP Port

When you have more than one OpenText eDirectory instance, the second instance and subsequent instances try to listen at the default 524 port instead of the NCP port on the loopback address.

To work around this issue, set the n4u.server.tcp-port parameter of the second instance to the port that it is supposed to listen on. The n4u.server.tcp-port parameter is located in the `nds.conf` file.

IMPORTANT: All OpenText eDirectory instances must be up before upgrading to OpenText eDirectory 9.0.

LDAP Transaction OIDs

In LDAP transaction support, the `supportedGroupingTypes` and `transactionGroupingType` OIDs are the same (`2.16.840.1.113719.1.27.103.7`).

Errors -5871 and -5875 in LDAP Trace

The -5871 and -5875 errors in LDAP trace are usually caused when LDAP client closes forcibly without doing an unbind. So, these errors need not be of concern and can be ignored. For more information on these errors, refer to the Error Codes Web site (http://www.novell.com/documentation/nwec/).

NDSCons Gives -625 Error if a Tree is Renamed

If you rename the tree on the primary server and shutdown the DHost on the secondary server, the NDSCons utility gives transport failure error message -625 on the secondary server while DHost keeps running on both primary and secondary servers. The error occurs because NDSCons was running on secondary server when the tree was renamed on the primary server. NDSCons works fine if you close it and then restart it.

NOTE: Tree rename is not a supported operation if you have EBA enabled servers in the tree.

Listening on Multiple NICs Slows Down OpenText eDirectory ldapsearch Performance

To work around this issue,

Disable the NICs in the configuration file that slow down the ldapsearch performance.

or

Enable Advanced Referral Costing (ARC) by using the `set NDSTRACE =!ARC1` command in DSTrace.

Unable to Limit the Number of Concurrent Users on Linux Platforms

You cannot limit the number of concurrent connections on Linux platforms. To resort to the old behavior (strict port-based checking), set following parameter in the `nds.conf` file.

```
n4u.server.mask-port-number=0
```

ndsd Fails to Shut Down Due to SLP

If you do not have an SLP Directory Agent (DA) configured on your network, finding services that use SLP may take a longer time. During OpenText eDirectory shutdown, ndsd tries to perform operations using SLP that may take a long time than the init script normally allows, thus causing a forced shutdown.

To workaround this issue:

1. Create an empty file with the name `hosts.nds` in the config directory. The config directory of a server can be obtained by running the following command `ndsconfig get n4u.server.confdir`
2. Set an environment variable NDS_USESLP to 0 by specifying export NDS_USESLP=0 in `/opt/novell/eDirectory/sbin/pre_ndsd_start`
3. Restart OpenText eDirectory.

Restarting NLDAP on Windows

After NLDAP is stopped, you need to restart the server to load NLDAP.

SecretStore over LDAP

The SecretStore functionality does not work over LDAP. To resolve this, you need to run the unloading: `$ nldap -u`, and then run loading: `nldap -l`.

Cannot Change the Passphrase after Unlocking SecretStore

SecretStore locks if you try to retrieve a forgotten password by logging in with user credentials and a wrong passphrase. You can unlock SecretStore with administrator rights, and the OpenText SecureLogin client allows you to log in without a passphrase. If you try changing the passphrase, the login fails and returns an error.

HTTP Server Uses SSL CertificateIP Even After it Has Expired

If you upgrade to OpenText eDirectory 9.0 from a lower version, the HTTP server continues to use the SSL CertificateIP even after the certificate has expired. This is because eDirectory 8.8 SP8 does not maintain SSL CertificateIP and does not reissue one even if the SSL CertificateIP expires or is deleted.

Hence, if the SSL CertificateIP expires or is deleted, you must manually create it by using the OpenText Identity Console or by using SSL CertificateDNS instead of SSL CertificateIP.

OpenText eDirectory Contains Two Different ldapsearch Binaries

Two sets of LDAP tools (`ldapadd`, `ldapconfig`, `ldapdelete`, `ldapmodify`, `ldapmodrdn`, and `ldapsearch`) exist on a SLES system (along with openldap2-client rpm) that has OpenText eDirectory installed: one in `/usr/bin`, installed by the SLES operating system and the other in `/opt/novell/eDirectory/bin`, installed by OpenText eDirectory.

Though the basic functionality of both sets of LDAP tools are the same, each set adds its own features on top of the basic functionality. Depending on the path settings in the PATH environment variable, the set of tools being used can differ and hence the features available can also differ.

ldapsearch Does Not Return Any Result

ldapsearch does not return any result when the bind user doesn't have the read rights for all the attributes that are part of the search filter.

To workaround this issue, ensure that the bind user has read rights for all the attributes that are part of the search filter.

Virtual List View displays an error message with OpenText eDirectory 9.1

Virtual list View (VLV) displays an error message with OpenText eDirectory 9.1 when all the partition replicas are not present within the OpenText eDirectory server where VLV is run.

Ensure that all the partition replicas are present within the OpenText eDirectory server where VLV is run.

OpenText eDirectory Doesn't Start After Moving the Datadir to a New Location

In case you move the `datadir` to a new location after configuring OpenText eDirectory on SLES 12 and above, ensure to perform the following steps:

- Update the new location of the `ndsd.pid` file in the service file found in the `/usr/lib/systemd/system/` location.

  For example, when the `nds.conf` file is originally located at the `/etc/opt/novell/eDirectory`, a sample service file will be created as shown below:

  `/usr/lib/systemd/system/ndsdtmpl-etc-opt-novell-eDirectory-conf-ds.conf@.service`.
- Re-load the daemon by using `systemctl daemon-reload` command.
- Restart the OpenText eDirectory server.

OpenText eDirectory Installation Fails Due to Restricted Execution Policy

OpenText eDirectory installation fails when the Windows execution policy is set to restricted for Powershell.

To workaround this issue, set the Windows execution policy to RemoteSigned for Powershell.

OpenText eDirectory Displays -659 Error Code While Performing LDAP Operations

While performing LDAP bind operation for the same user across multiple servers at the same time, the operation might be successful in one server and the synchronization happens to the next replica server immediately. But due to the mismatch in time stamp for the operation in different servers, login might fail displaying `-659` error code.

To fix this issue, set `NDSD_CC_SKULK_DELAY` environment variable to 5 or greater value. For more information, see "Synchronization Method" on page 117. If you still get the same error, set `NDSD_CC_SKULK_DELAY` to 5 or greater along with the newly introduced environment variable `NDSD_RETRY_MODIFY` to `true`.

NOTE: ◆Setting the above environment variable might impact OpenText eDirectory performance as the server will retry the same operation with a delay of 100ms up to a maximum of 2 seconds.

- OpenText eDirectory server should be stopped before setting the `NDSD_RETRY_MODIFY` environment variable. Restart the server once the environment variable is set.
- If you set the `NDSD_CC_SKULK_DELAY` environment variable to a value lesser than 5, `NDSD_RETRY_MODIFY` environment variable will not be effective.

Disabling Alert Reported Against the High-Valued Attributes

From OpenText eDirectory 9.2.5 onwards, when any of the high-valued attributes, namely `DirXML-EntitlementResult` and `oidpInstanceData`, exceed a default threshold value, OpenText eDirectory generates a `DSE_HIGH_VALUED_ATTR` event. This information is logged into the `hvAttr-alert.log` file located at:

- **Linux:** `/var/opt/novell/eDirectory/log`
- **Windows:** `C:\NetIQ\eDirectory`

The base threshold value for `DirXML-EntitlementResult` attribute is set to 5000. For every subsequent 500th value, the event is generated. In case of `oidpInstanceData` attribute, the base threshold value is 16KB. These values are set by default.

If you want to disable the alert on Linux platform, add the `NDSD_DISABLE_HIGHVALUED_ATTRIBUTES_ALERT` environment variable in the `env` file located at `/etc/opt/novell/eDirectory/conf` directory and set the value to `true`. On Windows platform, go to **Control Panel > System > Advanced System Settings > Environment Variables > System Variables > New** and add the new variable `NDSD_DISABLE_HIGHVALUED_ATTRIBUTES_ALERT` with the value as `true` and restart the system.

Handling Error -785 from Obituary Process

To workaround this issue, remove the `nsimAssignments` attribute from `CachedAttrsOnExtRefs` on each of the remote NCP Server objects, leaving it in the central servers.

Run `ndsrepair -R -I yes` on each server followed by triggering the Forcing the Backlink Process to Run (https://www.netiq.com/documentation/edirectory-92/edir_install/data/a7fcuas.html).

NOTE: This attribute will be added back when `ndsconfig` upgrade runs or in subsequent OpenText eDirectory updates.

Troubleshooting IPV6 Issues

Listeners for Unspecified IPv6 Addresses in Linux and Windows

A listener for an unspecified IPv6 address accepts both IPv4 and IPv6 connections on Linux. Because of this behavior, Linux does not allow you to start both IPv4 and IPv6 unspecified listeners for the same port at the same time. Therefore; if a listener is already configured for an unspecified IPv6 address, the listener on the unspecified IPv4 address fails to start. Linux uses an unspecified address for LDAP listeners.

On Windows, an unspecified IPv6 listener accepts only IPv6 connections. Therefore, you must configure a separate IPv4 listener to accept IPv4 connections along with IPv6 connections.

By default, both IPv4 and IPv6 listeners are configured for ldapInterfaces. Depending on the platform, ldapInterfaces starts the required listeners.

Troubleshooting EBA

EBA Based Login Fails If the User DN Contains an Attribute Without OID

EBA based login fails when the user DN contains an attribute without OID.

To workaround this issue, you must add the OID to the user DN attribute in the schema.