opentext™

OpenText™ Identity Manager CE 24.4 (v4.10)

Designer Administration Guide

January 2025

Legal Notice

Copyright 2009 - 2024 Open Text.

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 Creating a Project 19

Installing Designer . . . 19
Launching Designer . . . 19
When No Project Exists . . . 19
When You Want to Create an Additional Project . . . 20
When You Want to Import a Project . . . 22
When You Want to Disable a Project . . . 22

2 Creating a Model 23

Basic Tasks . . . 23
Accessing the Modeler . . . 24
Selecting a Modeling Mode . . . 24
- Developer Mode . . . 25
- Architect Mode . . . 27
- Dataflow Mode . . . 30
- Table Mode . . . 31

Working from the Palette . . . 32
- About the Palette . . . 32
- Palette Operations . . . 34
- Using Generic Applications . . . 34
- Fly-Out Palette . . . 35
- Resizing the Palette . . . 35
- Docking the Palette . . . 35
- Arranging Folders and Applications . . . 35
- Changing the Layout . . . 36
- Keyboard Support for the Palette . . . 37

Creating a Driver . . . 37
Copying and Pasting . . . 38
- Copying Applications . . . 38
- Copying a Driver Set . . . 39
- Copying an Identity Vault . . . 40
- Copying a Domain Group . . . 40
- Copying between Editors . . . 41

Moving Items . . . 42
In Line Editing . . . 42
Tooltips and Toolbar . . . 42
Organizing by Domain Groups . . . 43
- About Domain Groups . . . 44
- Key Features . . . 45
- Creating a Domain Group . . . 45
- Minimizing (Collapsing) Domain Groups . . . 47
- Restoring Domain Groups . . . 48
- Maximizing Domain Groups . . . 48

Using a List View of Domain Groups . . . 48
Auto-Placement of Neighbors . . . 49
Grouping into a New Domain Group . . . 49
Ungrouping a Domain Group . . . 51
Clearing Contents . . . 51
Changing a Domain Group Icon . . . 51
Keyboard Support for Domain Groups . . . 52
Connecting Applications . . . 52
Automatic Connections . . . 52
Connection Target Highlights . . . 53
Automatically Creating Objects . . . 53
Auto Redraw . . . 54
Manually Connecting . . . 54
eDir-to-eDir Connections . . . 54
Multiple Driver Connections . . . 55
Straightening Connections . . . 56
Reconnecting . . . 57
Driver Icons . . . 57
Selected Drivers . . . 58
Auto-Layout of Imported Objects . . . 58
Keyboard Support for Connections . . . 58
Aligning and Laying Out Components . . . 58
Alignment Hints . . . 59
Using Rulers . . . 60
Using a Grid . . . 62
Distributing Applications . . . 63
Auto-Layouts . . . 63
Layouts to Use for Imports . . . 64
Editing Multiple Objects . . . 64
Modeling Active Directory Domain Controllers . . . 64
Configuring a Connection . . . 64
Discovering Controllers . . . 65
Information about Domain Controllers . . . 65
Saving Your Model . . . 65

3 Configuring Objects in Designer 67

Viewing Object Properties . . . 67
Properties View . . . 68
Properties Dialog Box . . . 68
Operations Relating to Properties . . . 70
Configuring a Domain Group . . . 70
Configuring Identity Vaults . . . 70
Configuration . . . 71
Administrator . . . 73
Workflow Forms . . . 73
Packages . . . 74
Server List . . . 75
iManager . . . 75
Local Hostname . . . 75
Configuring Servers . . . 75
Configuring Driver Sets . . . 76
Driver Set General Options . . . 76
Driver Set Configuration . . . 77

Driver Set Global Configuration Values . . . 78
Java Environment Parameters . . . 78
Driver Set Log Levels . . . 79
Driver Set Named Passwords . . . 79
Driver Set Packages . . . 79
Driver Set Server List . . . 80
Driver Set Trace . . . 81
Configuring Libraries . . . 83
Configuring Drivers . . . 84
Driver General Settings . . . 84
Driver Configuration . . . 85
Engine Control Values . . . 87
Driver Global Configuration Values . . . 91
Driver Health Configuration . . . 93
Driver Log Level . . . 101
Driver Manifest . . . 102
Driver Named Passwords . . . 102
Driver Packages . . . 102
Reciprocal Attributes . . . 104
Driver Trace Levels . . . 107
Driver Icon . . . 110
Configuring Policies . . . 110
Editing a Policy Name . . . 110
Viewing References . . . 110
Configuring Resource Objects . . . 111
Configuring Categories . . . 111
Configuring Groups . . . 111
Configuring Packages . . . 111
Package General Settings . . . 112
Package Configuration Wizard . . . 112
Package Constraints . . . 113
Package Dependencies . . . 114
Package Initial Settings . . . 114
Package Languages . . . 114
Package License . . . 114
Package Linkage . . . 114
Package Readme . . . 115
Package Targets . . . 115
Package Vendor . . . 115
Configuring Package Content . . . 116
Package Content General Settings . . . 116
Package Content Installation . . . 116
Package Content Linkage . . . 116
Configuring Prompts . . . 116
Prompts General Settings . . . 117
Prompts . . . 117
Prompts Transformation . . . 117
Target Transformation . . . 118
Configuring Global Configuration Objects . . . 118
Global Configuration Object General Settings . . . 118
Global Configuration Object GCVs . . . 118
Configuring Jobs . . . 118
General . . . 118
Trace . . . 119

Configuring ID Policy Containers 119
Configuring ID Policies 120
Configuring a Notification Template 122
Configuring Application Properties 122
General 122
AD Domain 123
Administrator 123
Connectivity 123
Environment 127
Adding Prompts to a Driver Configuration File 127
Synchronizing Passwords 127

4 Managing OpenText Identity Manager Versions 129

Key Differences in OpenText Identity Manager Versions 130
Changing the OpenText Identity Manager Version 130
Tracking Versions of OpenText Identity Manager 131
Support for Driver Configuration Versions 132
Checking Projects for Version Issues 134
Adjusting the UI Based on the Version Number 134

5 Managing the Schema 137

Using the Manage Schema Tool 138
The Classes Tab 138
The Attributes Tab 142
Creating Classes and Attributes 148
Creating Identity Vault Classes 148
Creating Identity Vault Attributes 149
Modifying the Schema 150
Deleting Schema Definitions 150
Modifying Classes or Attributes 150
Renaming Schema Definitions 150
Deploying the Schema into the Identity Vault 151
Exporting the Schema to a File 154
Exporting the Schema to a .sch File 155
Exporting the Schema to an LDIF File 156
Importing the Schema 158
Importing the Schema from the Identity Vault 158
Importing the Schema from a File 162
Managing a Copy of an Application Schema 166
Editing an Application's Schema 166
Refreshing the Application Schema 167
Mapping Identity Vault to an LDAP Schema 167
Comparing the Schema 167

6 Understanding Packages 171

Using Packages 171
Advantages of Packages 172
Understanding Package Dependencies 173
Package Content 174

Installing or Upgrading Packages . . . 175
Installing Packages . . . 175
Adding Packages . . . 178
Upgrading Installed Packages . . . 179
Importing Packages into the Package Catalog . . . 180
Managing Installed Packages . . . 181
Customizing Default Packages . . . 183
Managing Package Versions Using Git . . . 184
Benefits of Using Git . . . 185
How Does Git Work? . . . 185
Adding an Existing Local Git Repository . . . 186
Cloning an Existing Local Git Repository . . . 186
Creating a Local Git Repository . . . 189
Removing or Downgrading Packages . . . 189
Uninstalling Packages . . . 190
Downgrading Installed Packages . . . 190
Removing Packages from the Package Catalog . . . 191
Running a Driver in Factory Mode . . . 191
De-activating Factory Mode . . . 192

7 Developing Packages 193

Why Use Custom Packages? . . . 193
Developing Custom Packages . . . 194
Preparing to Develop Packages . . . 195
Setting Default Package Preferences . . . 195
Creating a Development Driver . . . 195
Enabling Package Development Mode . . . 196
Defining Custom Package Structure . . . 196
Creating a Base Package . . . 197
Configuring Initial Settings . . . 199
Working with Package Prompts . . . 204
Understanding Package Prompts . . . 204
Understanding Package Prompt Types . . . 205
Understanding Package Prompt Transformations . . . 208
Example Default Prompt Transformations . . . 211
Example Default Target Transformations . . . 214
Examples of Modified Prompt Transformations . . . 220
Example of Modified Target Transformation . . . 221
Adding Default Package Prompts . . . 222
Creating Custom Package Prompts . . . 223
Editing Package Prompts . . . 225
Creating Identity Vault and Driver Set Packages . . . 226
Creating Libraries . . . 227
Adding GCV Resource Objects . . . 228
Adding Notification Templates . . . 228
Creating Feature Packages . . . 229
Configuring Mandatory and Optional Feature Packages . . . 230
Adding Content to Packages . . . 231
Adding GCVs to Feature Packages . . . 232
Adding Prompt Resources . . . 232
Adding Policies . . . 233
Adding Filter Extensions . . . 234

Adding Content to a User Application Driver 235
Modifying Content of a User Application Driver 235
Copying Packages 235
Building Packages 237
Versioning Packages 237
Localizing Packages 238
Adding and Configuring Licenses 239
Releasing and Publishing Packages 240
Best Practices for Package Development 241
Creating Packages 241
Naming Packages 241
Package Versioning 242
Defining Package Relationships 242
Adding Weights to Packages 242
Documenting Packages 243
Naming Package Items 243
Reusing Package Content 243

8 Managing the Flow of Data 245

The Dataflow View 245
Accessing the Dataflow View 247
Flow Arrows in the Modeler 247
Viewing How Attributes Are Synchronized 249
Changing the Data Flow 249
The Dataflow Editor 250
Filtering Views 251
Filtering Identity Vaults and Applications 252
Pinning the Identity Vault 253
Expanding and Collapsing the Identity Vault 254
Switching to an OpenText eDirectory Tree Icon 256
Viewing an eDir-to-eDir Driver 256
Keyboard Support 257
Adding Items in the Dataflow Editor 257
Adding an Identity Vault in the Dataflow Editor 257
Adding a Driver in the Dataflow Editor 257
Adding an Application in the Dataflow Editor 257
Adding Classes and Attributes 258
Adding Non-Filter Attributes 259
Removing Items from the Dataflow Editor 260
Removing an Identity Vault 260
Removing Classes and Attributes 260
Editing Items 260
Editing within the Dataflow Editor 261
Editing Non-Filter Attributes 262
Managing Schema 262
Removing a Flow 262
Changing How Data Flows 262
Generating HTML Reports 264
Integrating Passwords 265

9 Creating and Managing Policies 269

10 Setting Up E-Mail Notification Templates 271

Viewing Notification Templates . . . 271
Editing a Notification Template . . . 275
Selecting a Format . . . 276
Specifying a Subject . . . 276
Working with Tokens . . . 276
Attaching an Image . . . 277
Editing a Template Message . . . 278
Adding and Deploying a Notification Template . . . 278
Adding a Notification Template . . . 278
Importing a Notification Template . . . 279
Deploying a Notification Template . . . 280
Policy Builder and Notification Templates . . . 280
Configuring the E-Mail Server . . . 280

11 Importing into Designer 281

Importing Projects . . . 282
Importing a Project from the Identity Vault . . . 282
Importing a Project from the File System . . . 288
Importing a Project from iManager . . . 289
Importing a Project from a Version Control Server . . . 290
Importing a Library, a Driver Set, or a Driver from the Identity Vault . . . 292
Associating a Server to the Identity Vault . . . 292
Importing a Library from the Identity Vault . . . 293
Importing a Driver Set from the Identity Vault . . . 294
Importing a Driver from the Identity Vault . . . 298
Importing Packages . . . 301
Importing a Driver Configuration File . . . 302
Importing an OpenText Identity Manager Project from the File System . . . 303
Importing a Driver Configuration from a File in the Modeler View . . . 303
Importing from a File through the Outline View . . . 305
Importing Channels, Policies, and Schema Items from the Identity Vault . . . 307
Importing a Channel . . . 308
Importing a Policy . . . 310
Importing a Schema . . . 312
Using the Compare Feature When Importing . . . 315
Using Compare When Importing a Driver Object . . . 316
Using Compare on a Channel Object . . . 318
Using Compare on a Policy . . . 318
Matching Attributes with Designer Properties . . . 318
Error Messages and Solutions . . . 322

12 Documenting Projects 323

Creating a Document Style . . . 323
Editing a Document Style for Your Needs . . . 326
Editing a Style Template . . . 326
Editing Sections of a Style . . . 327
Generating a Document . . . 327

Using Your Style Template for Other Projects . . . 328
Documenting a Section of the Project . . . 329
Documenting Multiple Sections of the Project . . . 330
Customizing Styles to Include or Exclude Information . . . 330
Identity Vault Schema and Application Schema . . . 331
Using Project Configuration to Limit Information . . . 331
Advanced Editing of a Document Style . . . 333
What's In the Advanced Editing Mode . . . 333
A Walk-through Tutorial . . . 340
Selecting a Language for Generated Documents . . . 346
Double-Byte Font Support . . . 347

13 Using Entitlements 349

How Entitlements Work . . . 349
Designing Entitlements . . . 351
Terminology . . . 352
Entitlement Prerequisites . . . 352
OpenText Identity Manager Drivers with Preconfigurations that Support Entitlements . . . 352
Enabling Entitlements on OpenText Identity Manager Drivers . . . 353
Creating Entitlements through the Entitlement Wizard . . . 354
Valueless Entitlements . . . 355
Valued Entitlement that Queries an External Application . . . 357
Administrator-Defined Entitlements with Lists . . . 364
Administrator-Defined Entitlements without Lists . . . 366
Editing and Viewing Entitlements . . . 368
Entitlement XML Source and XML Tree Views . . . 369
Using the Entitlement DTD . . . 372
Managing Entitlements . . . 372

14 Scheduling Jobs 375

Job Scheduler Components . . . 375
Creating a Job . . . 376
Copying a Job . . . 377
Editing a Job . . . 377
Job Editor Selections on the General Settings Page . . . 378
Job Editor Selections on the Job Parameters Page . . . 381
Job Editor Selections on the Scheduler Page . . . 381
Job Editor Selections on the Notification Settings Page . . . 383

15 Deploying and Exporting 387

Preparing to Deploy . . . 387
Deploying a Project to an Identity Vault . . . 388
Deploying a Driver Set to an Identity Vault . . . 389
eDir-to-eDir Deployments and SSL/TLS . . . 390
Deploying a Driver to an Identity Vault . . . 390
Deploying a Channel to an Identity Vault . . . 392
Deploying a Policy to an Identity Vault . . . 393
Using the Compare Feature When Deploying . . . 394
Using Compare when Deploying a Driver Object . . . 394
Using Compare Before Deploying a Channel Object . . . 397

Using Compare Before Deploying a Policy . . . 397
Matching Attributes with Designer Properties . . . 397
Comparing Driver Set and Driver Attributes . . . 398
Renaming and Deleting Deployed Objects . . . 398
Troubleshooting Deployed Objects . . . 399
Exporting a Project . . . 399
Exporting to a File . . . 401
Using the Export Context Menu . . . 401
Exporting Configuration Files from the Modeler View . . . 402
Exporting Configuration Files from the Outline View . . . 402

16 The NetIQ XML Editor 405

About the NetIQ XML Editor . . . 405
Creating XML Files . . . 405
Validating Files . . . 410
Outline View . . . 410
XPath Navigator . . . 410
Using the Source Editor . . . 411
Using the Tree Editor . . . 413
Attaching a Schema or DTD . . . 415
Setting XML Editor Preferences . . . 416

17 Tools 417

Converting Earlier Projects . . . 417
Converting Earlier Projects . . . 418
Converting Projects with the Project Converter Wizard . . . 418
Migrating Driver Configuration Data to a New Server . . . 421
Using the Server Migration Wizard to Migrate the Driver Set . . . 421
Migrating a Driver Set to a Server in a Different Tree . . . 424
Migrating Server Data for Each Driver . . . 427
Opening a Web Browser . . . 428
Launching iManager . . . 428
Checking Your Projects . . . 429
Checking a Project . . . 429
Customizing the Project Checker . . . 431
Items That Are Checked . . . 433
Configuring TLS for eDir-to-eDir Drivers . . . 435
Prerequisites . . . 435
Enabling TLS . . . 435
Creating Certificates . . . 437
Using DS Trace . . . 438
Viewing DS Trace Live . . . 438
Creating a DS Trace Log File . . . 440
Viewing a DS Trace Log File . . . 440
Working with Generic Resources . . . 441
Creating a Generic Resource Object . . . 442
Editing a Generic Resource Object . . . 443
Updating Designer . . . 443

18 Version Control **445**

Installing a Subversion Server 446
Downloading and Installing the Server 446
Configuring the Server 446
Checking In a Project to a Version Control Server 447
Importing a Project from a Version Control Server 452
Accessing the Version Control View 455
Version Control Icons 456
Version Control View Headings 457
Version Control Options 458
Comparing Revisions and Resolving Conflicts 464
Comparing Revisions 464
Resolving Conflicts 466
The Modeler View Layout In a Team-Enabled Environment 467
Provisioning Objects 469
Version Control Best Practices 470
Best Practices 470
Managing Packages Best Practices 471
Best Practice Scenarios 472
Subversion and Version Control Interaction Rules 478

19 Setting Preferences **479**

Finding Preference Pages 479
General 480
Appearance 482
Compare/Patch 484
Content Types 485
Editors 486
Keys 488
Network Connections 488
Perspectives 489
Startup and Shutdown 489
Web Browser 490
Welcome 490
Help 491
Content 491
NetIQ 491
Designer 492
OpenText Identity Manager 495
Package Manager 511
Provisioning 514
Validation 518
Web 518
CSS Files 519
HTML Files 520
XML 522
XML Catalog 523
XML Files 524

20 Best Practices **529**

21 Troubleshooting Designer **531**

Running the Project Checker . . . 532
Viewing the Error Log . . . 532
Browsing the File System . . . 532
Using Menus . . . 532
Event Details . . . 533
Customizing Filter Settings . . . 533
Turning on Trace Messages . . . 534
Checking Loaded Plug-Ins . . . 535
Deploying OpenText Identity Manager Objects . . . 536
Deployment Considerations . . . 536
An Example Deployment Error . . . 538
Display Issues . . . 538
No F1 Help in Maximized Editors . . . 538
Using 120 DPI Fonts in Windows . . . 539
Display Issues on Linux . . . 539
Copying, Pasting, and Dragging in the Navigator View Don't Update Version Control . . . 540
Freeing Heap Memory . . . 540
Project Files Are Not Encrypted . . . 541
Users Cannot Import and Check In Multiple Instances of the Same Package Under Version Control . . . 541
Drivers Not Associated with Base Packages After Live Import . . . 541
Error Messages and Solutions . . . 543
Identity Vault Configuration Errors . . . 544
Driver Configuration Errors . . . 544
Internal Designer Errors . . . 545
eDirectory Access Errors . . . 546
OpenText eDirectory Object/Attribute Creation Errors . . . 547
Warnings . . . 549
Reporting Bugs and Giving Feedback . . . 554
Designer Does Not Deploy the Global Configuration Attribute on a Driver or a Driver Set . . . 555
Designer Takes Too Long To Deploy a Driver or a Driver Set . . . 555
Manually Removing Invalid Designer Shortcut from mac Launchpad . . . 555
Unable to Launch Designer Application on Mac . . . 556
Designer Fails to Deploy Driversets of Mixed Versions of OpenText Identity Manager . . . 556
Designer Does Not Respond If The Project Name Contains a Space . . . 556
Designer Fails on Linux to Open Data Item Mapping of a Workflow Activity Containing Non-ASCII Characters . . . 557
Importing a Workflow Created Using Older Version of User Application Driver . . . 557
Form Builder Does Not Launch Properly From Designer on Linux Platforms . . . 557
Unable to Deploy Large PRDs . . . 558
Unable to Connect to the Git Repository After Committing Package Changes for the Second Time . . . 558
Unable to Launch Form Builder on Linux Platforms . . . 559
Modifying The src Attribute Used in a Policy . . . 559
Modifying the do-create-resource Action Manually to Use REST API . . . 560

A Modeler Operations **561**

Modeler Space Operations . . . 561

Identity Vault Operations 563
Driver Set Operations 566
Driver Operations 569
Application Operations 573
Submenus 577
Keyboard Support 578

B Document Generator Core Support Templates 581

dgSection.xsl 581
dgFormat.xsl 582
idmConfig.xsl 585
idmUtil.xsl 586

C Adding Applications and Drivers to the Palette 589

Definition Folders and Files 590
Driver Configuration and Localization Files 590
Palette Folders and Files 590
The Notification Templates Folder 598
The Themes Folder 598
Adding to the Palette 599
Copying Configuration Files 599
Creating the Group 600
Adding a Key_Value Pair 600
Creating a Driver Definition 602
Creating the Application 606
Hooking Up the Custom Application 609
Protecting Your Customized Files 611

D Version Control with Subversion and Designer 613

Understanding Subversion 613
How Revisions Work In Subversion 613
Understanding Atomic Commits 615
Where Subversion Stores the Project Data 616
Moving an Existing Project 616
Administering Your Subversion Server 617
Server Specifications 617
Network Protocols 618
Authentication Schemes 621
Using Client Certificates 623
Configuring Subversion with Apache HTTP 624
Proxy Server Configuration 624
Subversion Server Backup 625
Taking Full Advantage of Version Control 626
When to Commit and When to Update 626
Comments 626
Creating and Using Tags 627
Subversion Keyword Substitution 627
Glossary 628

E Additional Driver Configuration 629

Configuring the Settings for the Multi-Domain Active Directory Driver 629
Adding Forests to the Multi-Domain Active Directory Driver 629
Configuring the Domain Connections 630
Configuring the Database Connections for the JDBC Driver 631

About this Book and the Library

The *Designer Administration Guide for OpenText™ Identity Manager* explains how to design, test, document, and deploy OpenText Identity Manager solutions in a highly productive environment. Newcomers can use wizards to build Identity Management solutions. Veterans and expert users can bypass the wizards and interact directly at any level of detail.

Intended Audience

This book provides information for individuals responsible for designing, deploying, and administering an OpenText Identity Manager solution.

Other Information in the Library

For more information about the library for OpenText Identity Manager, see the OpenText Identity Manager documentation website (https://www.netiq.com/documentation/identity-manager-4.10/).

1 Creating a Project

IMPORTANT: Projects created in Designer 4.6 or earlier must be converted to this version of Designer file format. When you open an earlier project in this version of Designer, it prompts you to convert the project to the new format.

- "Installing Designer" on page 19
- "Launching Designer" on page 19
- "When No Project Exists" on page 19
- "When You Want to Create an Additional Project" on page 20
- "When You Want to Import a Project" on page 22
- "When You Want to Disable a Project" on page 22

Installing Designer

To install Designer, see "Installing Designer" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Linux* or "Installing Designer" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Windows*.

Launching Designer

To launch Designer, perform the following steps:

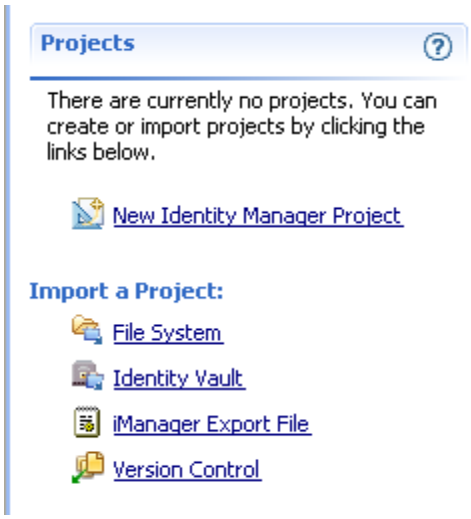
1. Navigate to the location on the server where you have installed Designer.
2. Launch Designer.
   - **Linux:** `./Startdesigner.sh`
   - **Windows:** **Designer for OpenText Identity Manager** application

When No Project Exists

1. Make sure that the Designer perspective (in the upper right corner) is selected.



2. If you are just starting Designer and have no projects in the Project tab, you see the following window:



3 Click **New OpenText Identity Manager Project** to launch the OpenText Identity Manager Project Wizard.

4 Name the project, then click **Finish**.

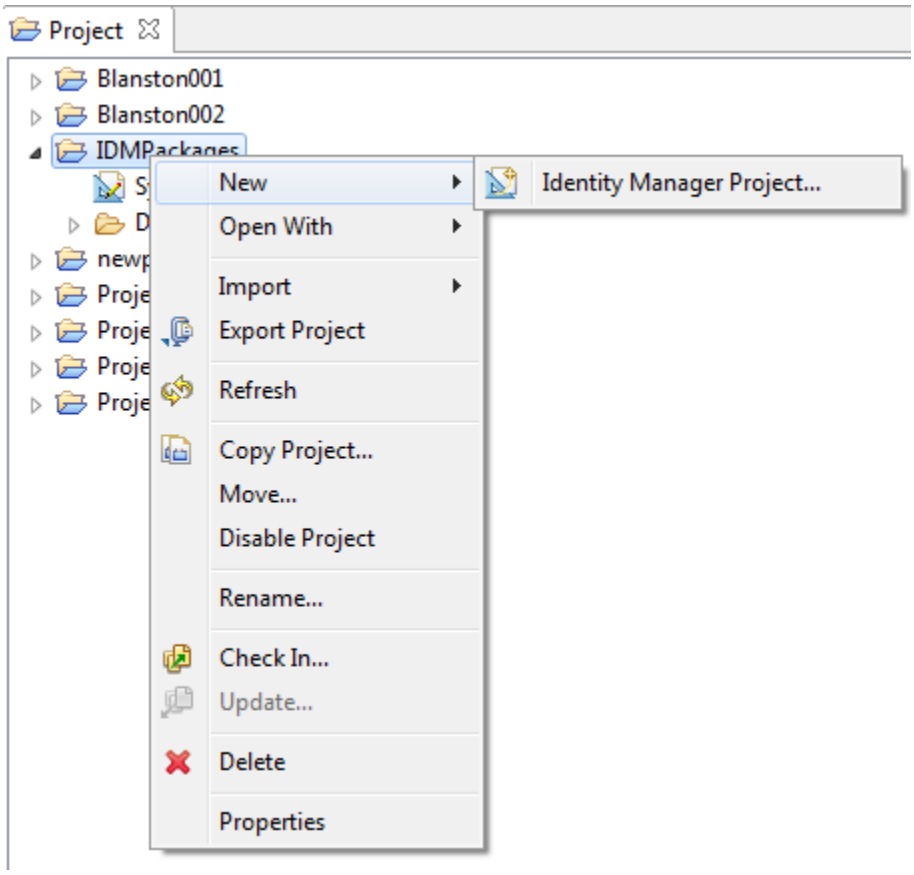
5 Select whether or not to import packages into the package catalog, then decide whether to allow Designer to always import package updates.

For more information about packages, see Chapter 6, “Understanding Packages,” on page 171.

6 (Conditional) If you selected to import packages, choose the packages you want to import, then click **OK** twice.

When You Want to Create an Additional Project

1 Right-click in the Project view pane, then click **New** > **OpenText Identity Manager Project**.



2 In the OpenText Identity Manager Project Wizard, name the project.

Designer stores the project in a local directory. You specified this directory when you installed Designer. Typically, this default directory is `$HOME/designer_workspace` for Linux and `%UserProfile%\designer_workspace` for Windows. To specify a different directory, deselect **Use Default**, then browse to and select the desired directory.

WARNING: Earlier Designer workspaces are not compatible with this version of Designer. You need to point this version of Designer to a new workspace, and not to a workspace used by a previous version of Designer.

If you have an earlier project, you can import the project into this version of Designer (**File > Import > Project from File System**). Be sure **Copy project into the workspace** is selected. Importing the project runs the Converter Wizard, making the project compatible with Designer architecture and placing it under your designated Designer workspace directory (`designer_workspace` by default).

---

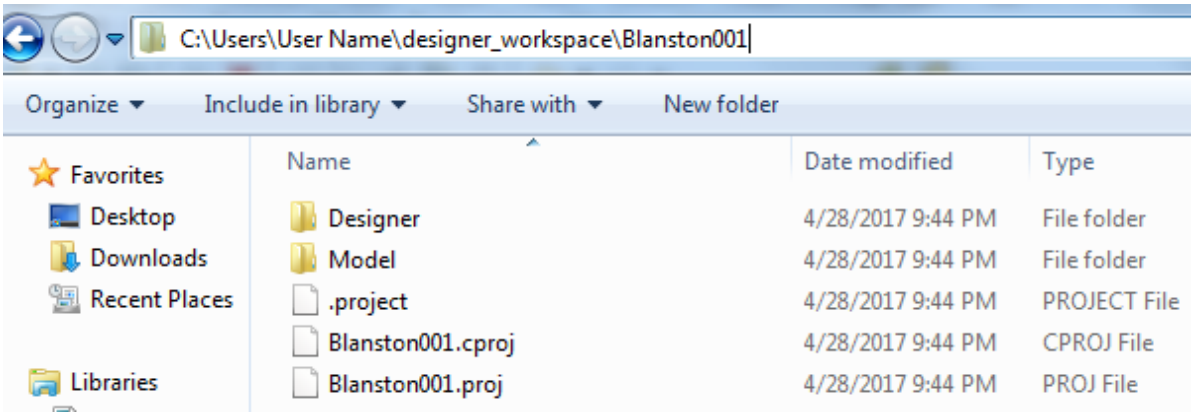
3 Click **Finish**.

4 Select whether or not to import packages into the package catalog, then decide whether to allow Designer to always import package updates.

For more information about packages, see Chapter 6, “Understanding Packages,” on page 171.

5 (Conditional) If you selected to import packages, choose the packages you want to import, then click **OK** twice.

The project is stored in a directory structure with the project name as the initial directory containing files with a `.proj` and a `.project` extension. For example, the project is stored in the `c:\Users\User Name\designer_workspace\Blanston001` directory on a Windows computer.



The project name appears in the Project view. When you select the System Model icon under the project name, Designer opens the Modeler (an editor) for the new project.

For information about saving a project, see "“Converting Earlier Projects” on page 417".

When You Want to Import a Project

To import a project from an Identity Vault or from the File System, see Chapter 11, “Importing into Designer,” on page 281.

IMPORTANT: You can open projects updated in Designer 4.x, especially the ones where the linkages were migrated, with earlier versions of Designer.

When You Want to Disable a Project

You can disable and enable projects from the Project view.

1. To disable a project, right-click a project in the Project view and select **Disable Project**.

   When a project is disabled, it is not accessible from any of the other views, including the Version Control view, and the project is converted to a placeholder in the Project view.

2. To enable the project, right-click the project placeholder in the Project view and select **Enable Project**. The project is again accessible in the other views.

2 Creating a Model

The Designer Modeler lets you create and manipulate a model of your OpenText Identity Manager environment within a Designer project.

Basic Tasks

You need to perform several basic tasks for creating a model after you have created a project.

1. In Designer, select a project.

   If your project does not appear in the Modeler, open the Project view (**Window** > **Show View** > **Project**), expand the project, then double-click **System Model**.

2. Drag an Identity Vault object from the palette to the Modeler.

   When you create an Identity Vault or server in Designer the default OpenText Identity Manager engine version is the same. Designer assumes that the Identity Vault has capabilities. You can successfully deploy and run projects only on OpenText Identity Manager servers.

   You can easily change the engine version by selecting a version from the **Server DN** field. However, selecting earlier engine versions removes any later version capabilities and features from within Designer.

   Before you deploy a project, you must associate a server with the Identity Vault. You do this through the Identity Vault properties. See “Configuring Identity Vaults” on page 70.

   You can add multiple Identity Vaults.

3. Configure a driver set.

   Each Identity Vault contains a driver set. See “Configuring Driver Sets” on page 76.

4. Add applications.

   Drag applications from the palette to the Modeler view. See “Configuring Application Properties” on page 122.

5. Create or configure drivers.

   Driver connections are automatically drawn between the application and the driver set. See “Creating a Driver” on page 37 or “Configuring Drivers” on page 84.

6. Develop and customize your model.

   Develop according to what you planned in “Planning an Identity Project” in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

7. Save your model (design).

   Do one of the following:

   - From the main menu, select **File** > **Save** (or **Save All**).
   - From the main menu, select **File** > **Close** > **Yes**.
   - Click the **X** in the Modeler’s tab, then select **Yes**.

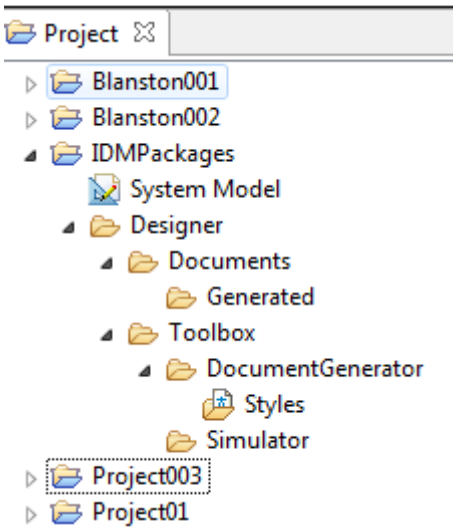
Accessing the Modeler

The Modeler space is the main working area. It is an editor where you design projects. It is the main workspace and primary means of interacting with Designer. All other editors, views, and dialog boxes support and provide functionality for the Modeler.

To get started, you create a project and drag items from the palette into the Modeler space. Then you arrange and configure the items.

If the Modeler does not display:

1 Expand a project in the Project view.



If you have not yet created a project, create one.

2 Double-click **System Model**.

Selecting a Modeling Mode

The Modeler has tabs along the bottom, so that you can switch among different modeling modes. The modes have different advantages, depending on the task you're trying to do and the role that you are acting in.

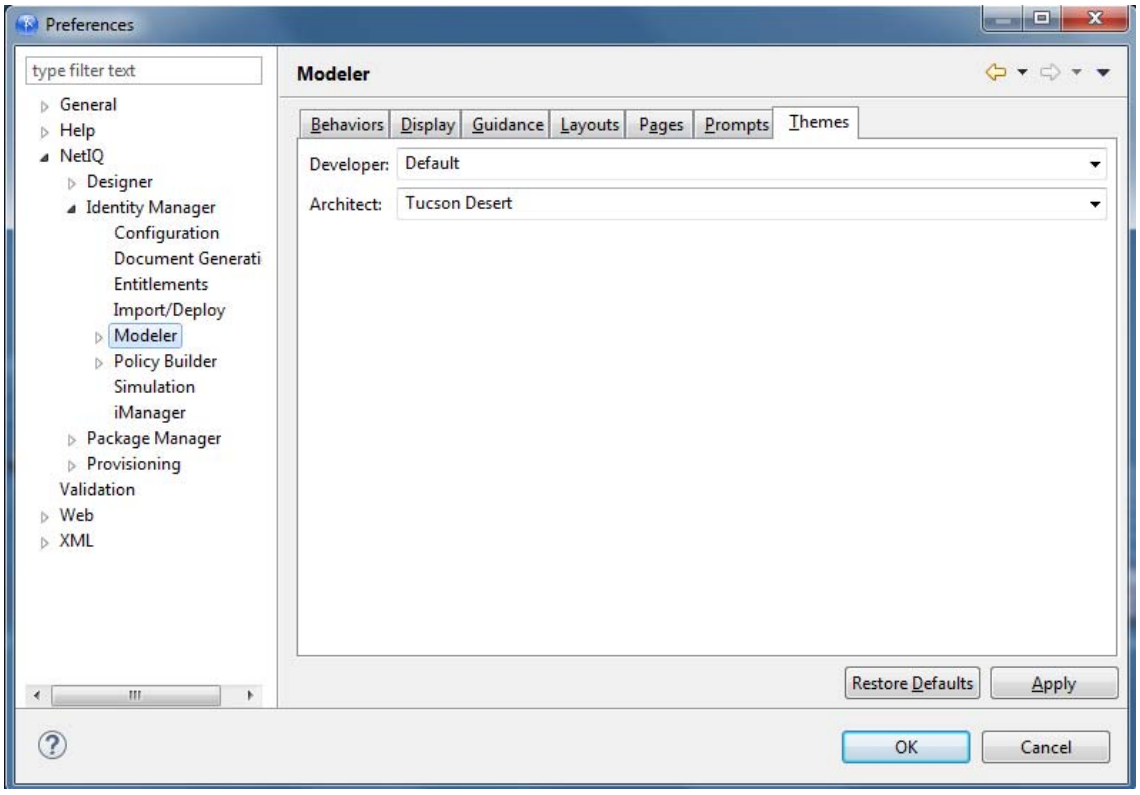


The modes are synchronized with each other with selection, data, and content. They are also synchronized with the Outline view and Thumbnail view.

As you switch modes in the Modeler editor, the editor tab at the top displays the mode that you are in as you switch modes, Designer also remembers and restores to the Modeler page you were last on when you close and re-open a project. This helps you return to the last mode you were in.

By default, the theme preference is different for each mode. You can configure each theme independently in the Modeler preferences:

1 Click **Window** > **Preferences,** then select **NetIQ** > **Identity Manager** > **Modeler**.

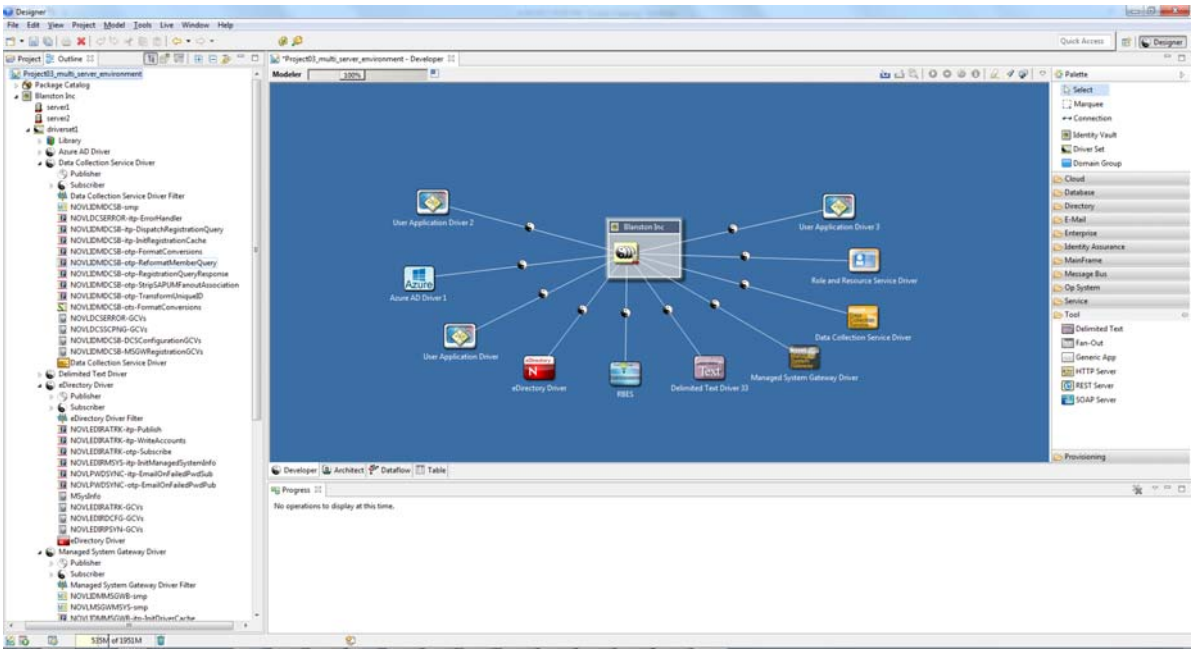


2 Click the **Theme**s tab.

3 Select a theme, then click **OK**.

Developer Mode

Use Developer mode to do all low-level operations with driver sets, drivers, policies, and applications.



This mode lets you manage all of the visual elements and configuration details that you need to fully build and deploy an identity solution.

In Developer mode, the palette organizes the applications and systems into categories. You can customize them to display as one alphabetical list by using the Modeler Preferences. See “Palette Page” on page 507.

Working with Labels

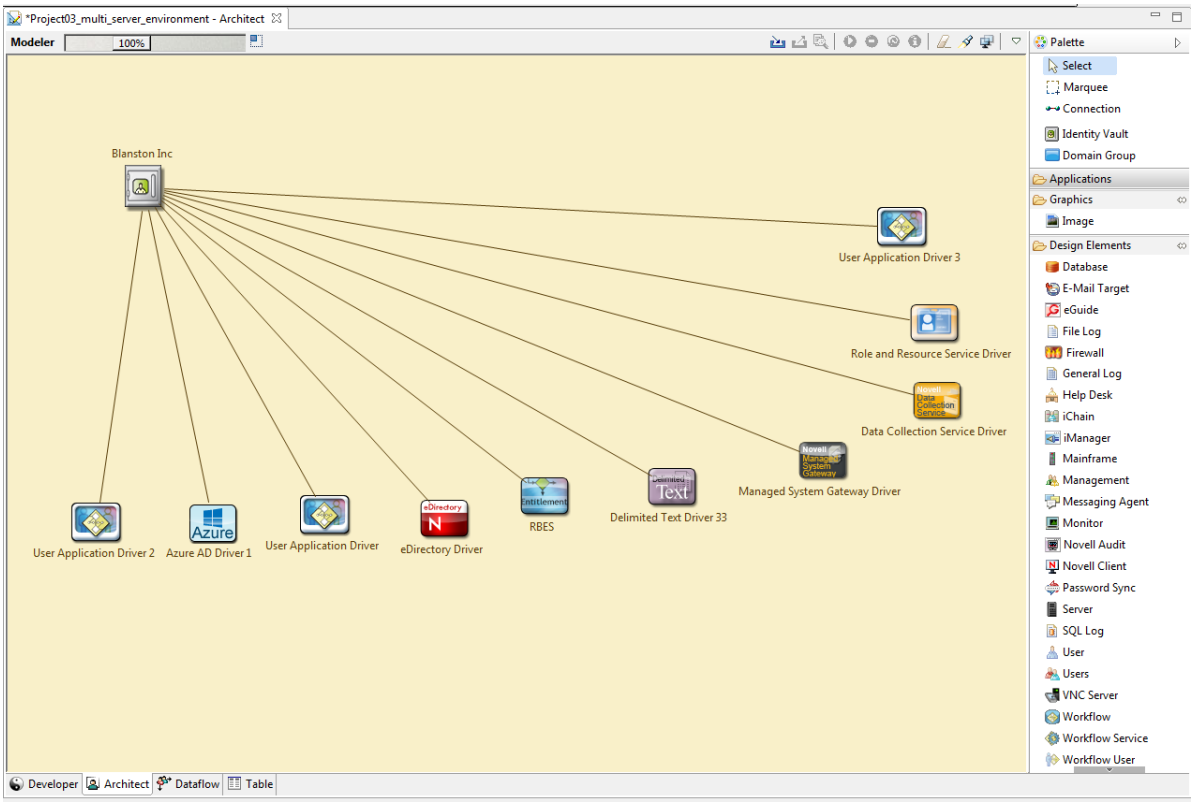
Figure 2-1 *An Application’s Label*



By default in both Developer and Architect modes, labels appear under application icons in the Modeler. They also appear above Identity Vaults in Architect mode.To configure these labels to not appear, use the Modeler Preferences. See “Modeler” on page 505.

Architect Mode

Figure 2-2 *Architect Mode in Designer*



Use the Architect mode to work at a design level for your projects. Because the design level does not show drivers, driver sets, or policies, you focus more on systems. This mode helps you do large-scale design, which is more intuitive to architects and business strategists.

It is quite likely that you will start in this mode when you begin each project. You will probably spend time putting together an accurate diagram of your enterprise as you consult with various people throughout your organization. As you do so, you should capture key information about each system, such as the owner, contact information, machine environment, software versions, and authentication credentials. As you go through this process, you will also define your project requirements, start thinking about your data, and capture that information in your project.

When the time is right, you can switch to the Developer mode and delve into the technical details of building a working solution. Depending on the size of your project and the makeup of your team, you could have architects and designers build high-level solutions with Designer in the Architect mode, and then send the project to identity developers who understand the details of writing policies and configuring systems. They can share the same project.

In Architect mode, you can connect any design element with any other design element, application, image, or Identity Vault. The connecting lines enable you to express any relationship, making Architect mode a general-purpose, high-level business model. The Architect-mode lines don’t display when you switch to the Developer mode.

NOTE: When you add icons representing driver applications through the Architect mode, you need to configure those drivers in the Developer mode. When you have added the necessary drivers and switch to the **Developer** tab, right-click the line between the driver icon and the driver set, then select **Run Configuration Wizard**.

The design elements have connectivity information tied to them. You can use design elements to perform live operations or to remotely control other elements that are in your environment but are not necessarily included in your OpenText Identity Manager infrastructure.

When using the Architect mode, you should be familiar with the following:

- "The Palette in Architect Mode" on page 28
- "High-Level Data Flows in Architect Mode" on page 28
- "Tasks" on page 29

The Palette in Architect Mode

In Architect mode, the palette lists all applications in one folder and design elements in another folder. The Architect Modeler view now contains all of the graphical modeling tools that are present in the Developer Modeler view. This includes:

- Rulers
- Snap-in guides
- Alignment hints
- Grid
- Snap-in movement

The Graphics folder has an **Image** icon. When you drag this icon to the Modeler, Designer displays a generic graphic:

To edit the properties of this icon:

1. Right-click the icon, then select **Properties**.
2. In the **Name** field, replace **Image** with a caption.
3. Browse to and select a replacement graphic, then click **OK**.

   You might need to reduce the size of the graphic before importing it.

After the image is in the Modeler, you can drag it, change it, connect lines to it, align or distribute it, or delete it.

High-Level Data Flows in Architect Mode

To set data flows in Architect mode:

1. Right-click the line between an application and an Identity Vault.
2. Select **Show Dataflow View**.
3. Right-click the line again and select **Dataflow**.
4. Specify synchronization and notification events, then click **OK**.

This option is used the same way as in Developer mode except that in Architect mode, Designer automatically configures all the details (schema, filters, and mapping policies) for you. You won't see the Data Flow Wizard for these details. Before deployment, you can edit the details by using Developer mode.

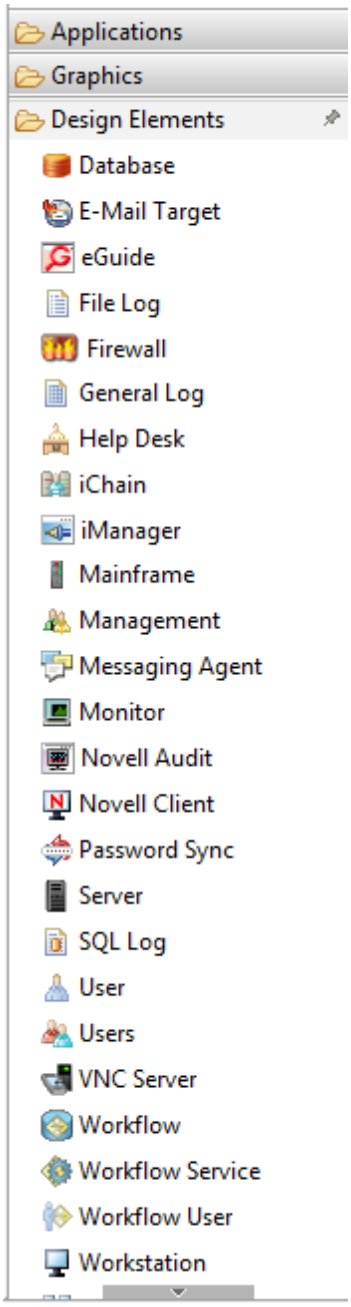
Tasks

You can perform the following tasks in Architect mode:

- Straighten connections (edges). See "Aligning and Laying Out Components" on page 58.
- View Password Sync icons and edit synchronization. See "Integrating Passwords" on page 265.
- Auto-connect eDir-to-eDir.
- When deleting the driver line, view a prompt to confirm drivers being deleted.
- Display design elements in your model.

  Open the **Design Elements** folder on the palette, drag design elements onto the Modeler, and connect the design elements.

Figure 2-3 *Items in the Design Elements Folder*



Dataflow Mode

The Dataflow mode launches the Dataflow editor, so that you can see all of the filters that control how data flows between the managed systems and Identity Vaults. In the Dataflow editor, you can right-click an eDir-to-eDir connection and have the option to remove the connection.

The Dataflow mode is synchronized with the Modeler and with the Outline view when you add, delete, change, or synchronize objects. Also, you can see how passwords flow from each server. See Chapter 8, “Managing the Flow of Data,” on page 245.

The Dataflow toolbar enables you to perform the following actions:

- Deploy driver filters for all drivers in the Dataflow view.
- Refresh the Dataflow view's UI screen.
- Save the current Dataflow view to an HTML file. You can select the directory where you want to save the file.
- Save all of the filtered views (Notify, Sync, Reset, Password Sync) to an HTML files. You can select the directory where you want to save the files.
- Go up and down to the Identity Vaults.
- Create a new Identity Vault.
- Add an application driver for a managed system.
- Filter Identity Vaults and application drivers out of the Dataflow view.

The pull-down menu allows you to perform the following:

- Expand all containers
- Collapse all containers
- Launch Dataflow preferences
- Get help

The Architect and Modeler views contain the same pull-down menu with the same functionality.

Table Mode

Figure 2-4 *Global Table Editor*

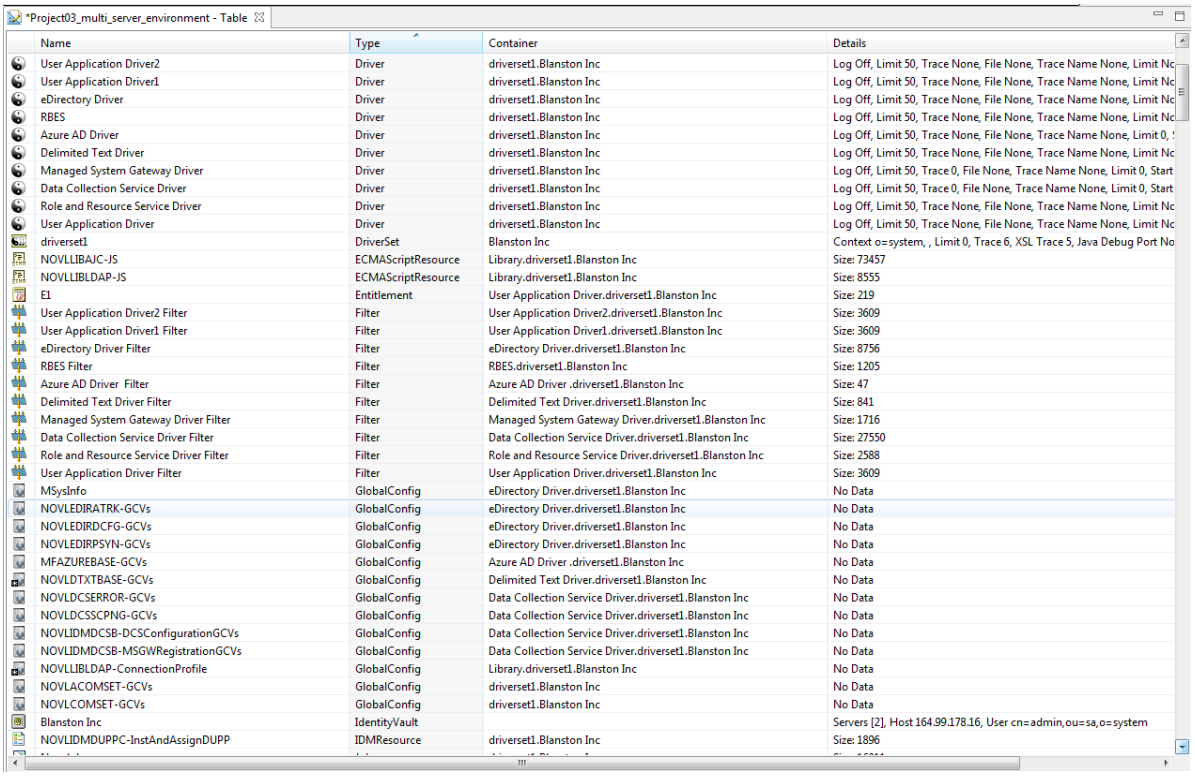


Table mode provides a Global Table editor, which lists all design elements in the project. You can scroll through this table to quickly scan essential information, such as the element's type, the container where the element resides, and details, such as an element's size, or driver and server information. You can efficiently find all items of a particular type and edit their settings.

To edit an entry in the table, double-click a line, or right-click a line and select **Open With**, then select an editor. You can also right-click a line, select **Open**, and Designer launches the editor that has been associated with the action. For example, drivers open their Properties page, and policies open in the Policy Builder.

When you select an entry in the table, Designer synchronizes the selection with the Outline view, so that you can view the selection's container.

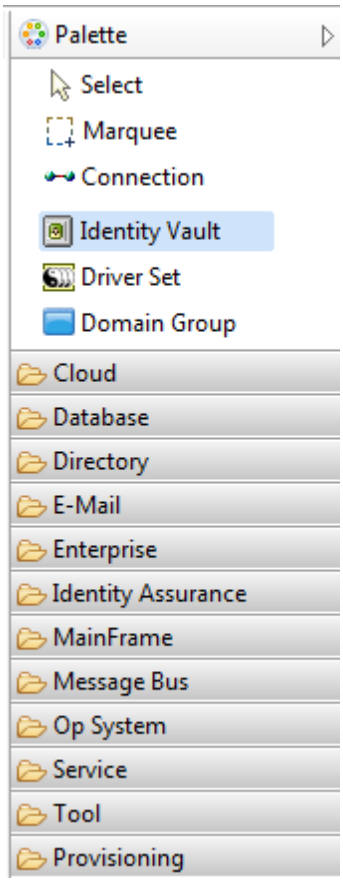
To sort the lists, click a column header.

Working from the Palette

- "About the Palette" on page 32
- "Palette Operations" on page 34
- "Using Generic Applications" on page 34
- "Fly-Out Palette" on page 35
- "Resizing the Palette" on page 35
- "Docking the Palette" on page 35
- "Arranging Folders and Applications" on page 35
- "Changing the Layout" on page 36
- "Keyboard Support for the Palette" on page 37

About the Palette

The palette is the source of all of the items that you add into the Modeler.



To build a model, do one of the following:

- Drag and drop items from the palette to the Modeler space. When you drag and drop an application, it auto-connects to the closest driver set.
- Click an item in the palette, then click in the Modeler space where you want the item to go.

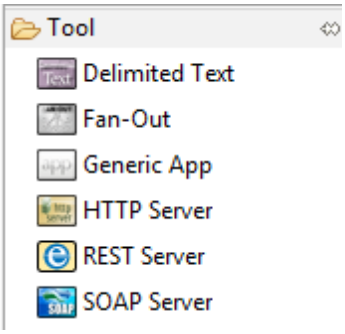
Palette Operations

Table 2-1 *Palette Operations in Designer*

Operation	Description
Connection	Connects items in the Modeler space.
Identity Vault	Places an Identity Vault in the Modeler space.
Driver Set	Places an OpenText eDirectory Driver Set object in an Identity Vault. All applications that you want to connect use a Driver Set object as a hub between the two applications.
Domain Group	Lets you group and organize items in the Modeler space.
Folders	Applications are organized within folders or drawers. To open or close a folder, click it. To hold the folder in place and make sure that it does not fully collapse (even when you open other folders), click the pin. When the Palette is full, unpinned folders automatically close when you open another folder.
Applications	The various applications that you can connect are grouped into folders by type. You can drag and drop these applications to the Modeler space and begin editing them. The Modeler automatically adds a connecting line, which represents a driver.
Scrolling Arrows	Small directional arrows. If a folder has many items, or if the screen area is restricted, scrolling arrows appear. To scroll through he contents of a folder, click the arrows.

Using Generic Applications

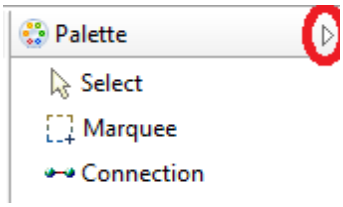
Figure 2-5 *The Generic App Option on the Palette*



Scenario: A Generic Application. Fridrik creates a project with his own items and graphics, in his own version of Designer. He transfers the project to you, but you are using a different version of Designer, which does not understand those items. Your version renders the transferred objects as Generic applications.

Fly-Out Palette

Figure 2-6 *The Palette's Control Arrow*



Designer provides a small control arrow on the palette. Click the arrow to open or collapse the palette.

To temporarily open the palette again, hover the cursor over the collapsed palette, below the control arrow. The palette quickly expands. This is fly-out mode.

To change the palette from fly-out mode, click the control arrow again. The state persists and is restored the next time you run the application.

Resizing the Palette

1. Click the palette's thick border that faces the Modeler space.
2. Drag the line.

   The size persists and is restored the next time you run the application.

Docking the Palette

To dock the palette on the left or right of the Modeler space:

1. Click the top palette header.
2. Drag the palette to the desired location.

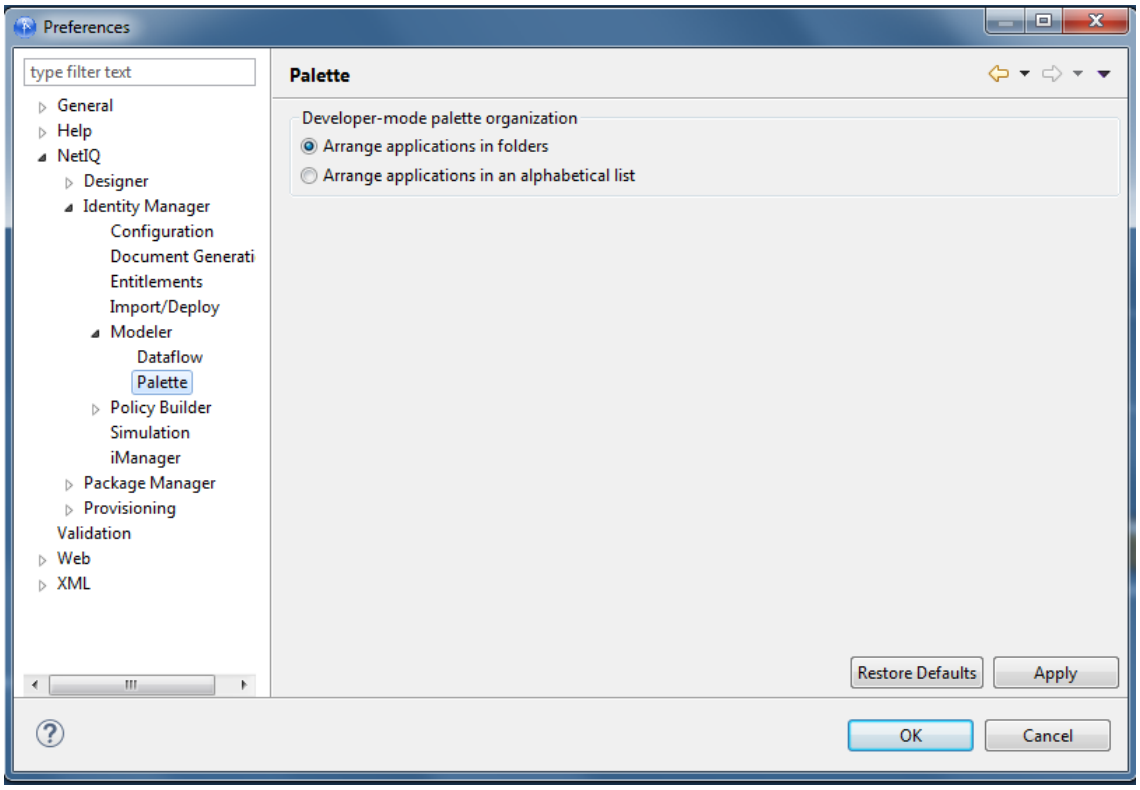
   The location persists and is restored the next time you run the application.

Arranging Folders and Applications

By default, applications are placed in folders.

To arrange applications alphabetically instead of in folders:

1. Click **Window** > **Preferences** > **NetIQ** > **Identity Manager**> **Modeler** > **Palette**.



2 Select **Arrange applications in alphabetical list**, then click **OK**.

Changing the Layout

1 Right-click the palette.

2 Select **Layout**.

3 Select an option.

Setting	Description
Layout: Columns	Displays folders and applications in columns.
Layout: List	Arranges folders and applications in a list.
Layout: Icons Only	Removes descriptive labels.
Layout: Details	Briefly describes palette items.
Use Large Icons	Toggles the size of icons used for applications.
Settings	Enables you to set the layout and icon size in one dialog box. Controls how folders (drawers) behave.

Keyboard Support for the Palette

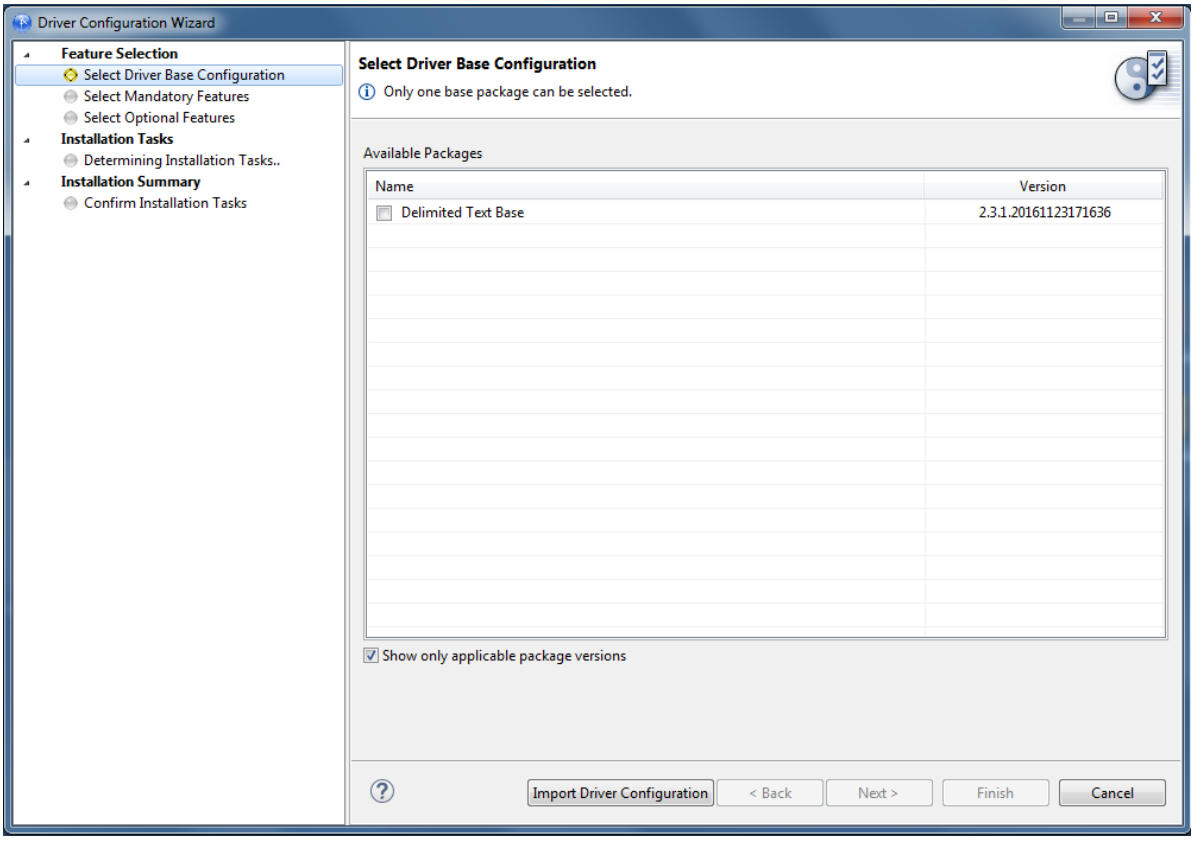
Table 2-2 *Shortcut Keys for the Palette*

Keystroke	Description
Left-arrow	Collapses an open folder. The focus must be on the folder, not the application.
Right-arrow	Opens a collapsed folder. Moves into an open folder.
Up-arrow	Moves up to the next folder.
Down-arrow	Moves down to the next folder.

Creating a Driver

Drivers connect the applications to the Identity Vault and provide the means for the data to synchronize. To create a driver, select an application from the palette, then drag and drop it on the Modeler. The application is connected to the closet driver set and the Driver Configuration Wizard launches.

Figure 2-7 *Driver Configuration Wizard*



The purpose of the Driver Configuration Wizard is to help you install drivers. In the past, that meant walking through the import of a driver configuration file. Now, the Driver Configuration Wizard walks you through installing packages or driver configuration files. However, only packages contain new driver content. The driver configuration files are not updated from this point on.

To create a driver with packages, select the available base package listed. If there are no packages listed, then the packages are not imported into the package catalog. For more information about importing and installing packages, see "Installing or Upgrading Packages" on page 175.

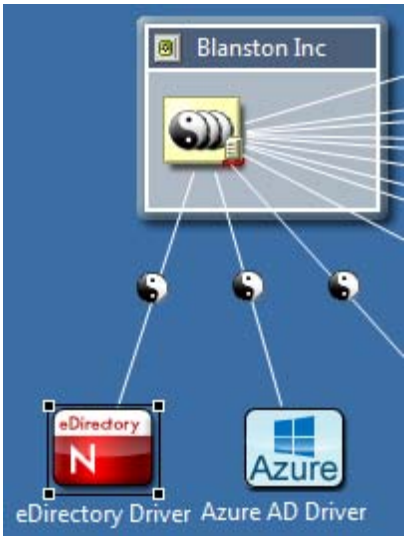
To create a driver with a driver configuration file, click **Import Driver Configuration**. All of the driver configurations files for the version of your OpenText Identity Manager server are listed. For more information about importing a driver configuration file, see "Importing a Driver Configuration File" on page 302.

Copying and Pasting

- "Copying Applications" on page 38
- "Copying a Driver Set" on page 39
- "Copying an Identity Vault" on page 40
- "Copying a Domain Group" on page 40
- "Copying between Editors" on page 41

Copying Applications

Figure 2-8 *Applications to Copy*



You can copy and paste the following items within the same editor or to another editor:

- Applications, including custom applications

- Disconnected applications
- Driver icons

1. Select an application or driver icon.
2. Press Ctrl+C, then Ctrl+V.

The copy and paste operations are also accessible from the Clipboard context menu. (Currently, they aren't accessible from the main menus.)

When you copy an application in the same editor, Designer copies all of the application's attributes, and copies all sub-elements. Therefore, all drivers that the application is connected to are copied, and all policies that the drivers contain are also copied. The new application connects to the same driver sets that the previous application connected to.

To copy an application to a different driver set (in the same editor or in another editor):

1. Select the application.
2. Press Ctrl+C.
3. Select the target driver set that the application connects to.
4. Press Ctrl+V.

If you copy and paste an application without selecting a target driver set, Designer makes a copy and connects it to the current driver set.

You can select multiple applications and then copy and paste them.

Copying a Driver Set

You can copy and paste driver sets within the same Identity Vault or to another Identity Vault in the same editor or in another editor.

1. Select a driver set.
2. Press Ctrl+C, then Ctrl+V.

When you copy a driver set in the same editor, Designer copies all of the attributes of the driver set, including the following:

- All drivers that the driver set is connected to
- All policies that the drivers contain
- All target applications

To copy to a different editor:

1. Select a driver set.
2. Press Ctrl+C.
3. Select the target Identity Vault in the other Modeler editor where you want the driver set to be copied to.
4. Press Ctrl+V.

By default, the new driver set is created in the same Identity Vault as the one that it was copied from. However, if you select another Identity Vault, the driver set is copied there.

After you copy and paste, you might need to move the pasted objects to a better location so that they don't cover up an existing object. To do this, leave the objects selected after you paste them, then move them. Or, use the following procedure to easily select objects:

1. Right-click a driver set.
2. Click **Select All Connected Applications**.
3. Move one of the selected applications.

   All connected applications move together.

When you copy a driver set, it has the same settings, except for the selected servers, which are blank. This exception occurs because the OpenText Identity Manager engine does not allow more than one driver set on an Identity Vault to be associated with the same server. Therefore, you need to set up the servers for the new driver set. If you copy an Identity Vault, Designer copies the driver sets. The new driver set has the same server settings set up for you.

You can select multiple driver sets and then copy and paste them. To copy and paste multi-driver connections, you must copy the driver set or Identity Vault that contains them.

Copying an Identity Vault

You can copy and paste Identity Vaults within the same editor, to another editor in the same Modeler space, or in a specific Domain Group.

1. Select an Identity Vault.
2. Press Ctrl+C.
3. Select nothing or select the target Domain Group (in the same editor or another) where you want the Identity Vault to be copied to.

   If you select nothing, the new Identity Vault is copied to the right of the previous Identity Vault in the current editor.
4. Press Ctrl+V.

The new Identity Vault appears to the right of the previous Identity Vault and is the same size as the one that it is being copied from.

When you copy an Identity Vault, Designer copies all of the elements of the Identity Vault. The elements include servers, e-mail templates, driver sets, and connected applications.

You can select multiple Identity Vaults and then copy and paste them.

Copying a Domain Group

You can copy and paste Domain Groups within the same editor, to another editor in the same Modeler space, or in a specific Domain Group.

1. Select a Domain Group.
2. Press Ctrl+C.
3. Select the location for the new Domain Group.

If you select nothing, the new Domain Group is copied to the right of the previous Domain Group in the current editor.

4 Press Ctrl+V.

The new Domain Group appears to the right of the previous Domain Group, and is the same size as the one it was copied from.

When you copy a Domain Group in the same editor, Designer copies all of the attributes of the Domain Group. However, Designer doesn't copy all sub-elements.

You can select multiple Domain Groups and then copy and paste them.

Copying between Editors

To easily copy and paste between two editors:

1 Using the Project view, open two projects.

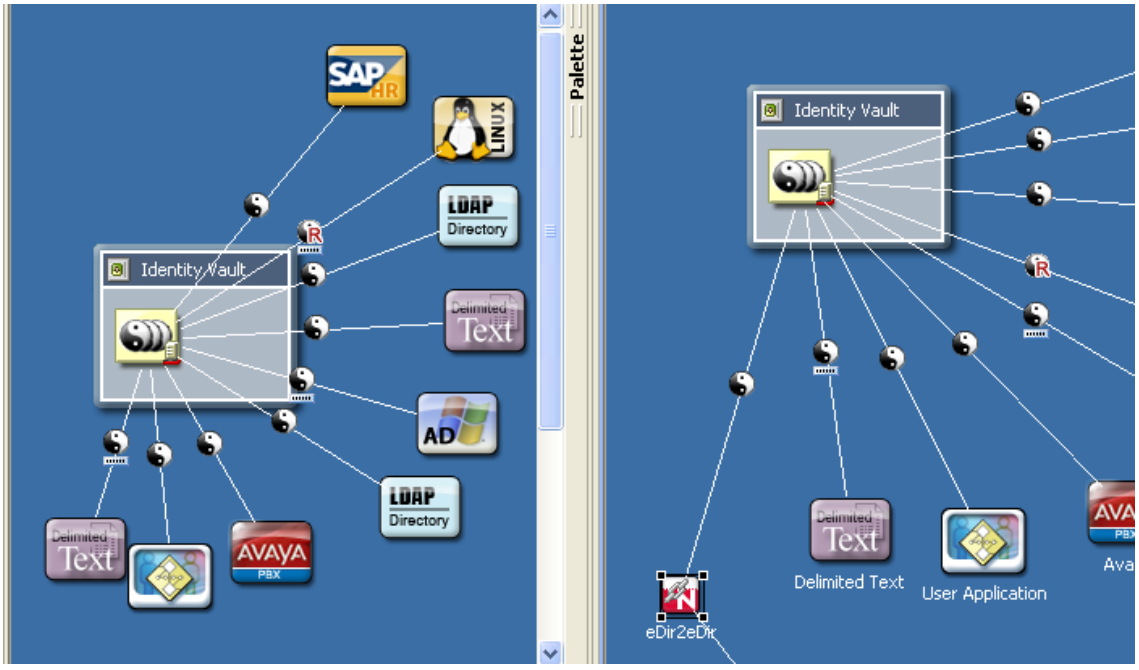
One project is active. The second project's tab displays at the top of the Modeler.

2 Close the palette by clicking the control arrow on the palette's title bar.

3 Click the second project's tab and drag it to the Modeler's right border.

The tab changes to a folder icon until it arrives near the border, where the folder changes to an arrow.

4 Release the mouse button.



5 Copy items from one editor to the other.

Moving Items

After an item is in the Modeler space, you can move it by dragging it to a new location. The Modeler prevents you from placing objects where they do not belong. For example, you cannot move a driver set out of an Identity Vault to the Modeling space, or drop an application inside of an Identity Vault. You can always drag objects into a Domain Group, or drag a driver set from one vault into another.

If you drag a driver set into an Identity Vault, the Identity Vault automatically grows or shrinks to fit the driver set, so you don't need to manually resize the vault. This behavior can be turned on or off in Preferences. See "Modeler" on page 505.

In Line Editing

Figure 2-9 *An In Line Edit*



To edit the names of objects, do one of the following:

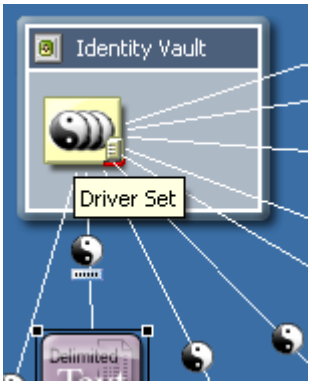
- Select the item, press F2, then edit the label.
- Double-click the item, then edit the **Name** field.

You can do an in line edit for any type of item in the Modeler, including the driver lines.

Tooltips and Toolbar

As you mouse over objects in the Modeler, a tooltip appears with the name of the object.

Figure 2-10 *A Tooltip*



The Modeler also provides a toolbar.

Figure 2-11 *The Modeler Toolbar*



The Modeler toolbar enables you to quickly find often-used features:

- Search
- Find a driver's status (also available from the **Live** menu when you select a driver set or Identity Vault)
- Start, stop, or restart a driver (also available from the **Live** menu when you select a driver set or Identity Vault)
- Clear all items
- Save a snapshot of the model

The drop-down menu allows you to perform the following:

- Expand all containers
- Collapse all containers
- Launch Modeler preferences
- View demos on how to use the Designer
- Get help

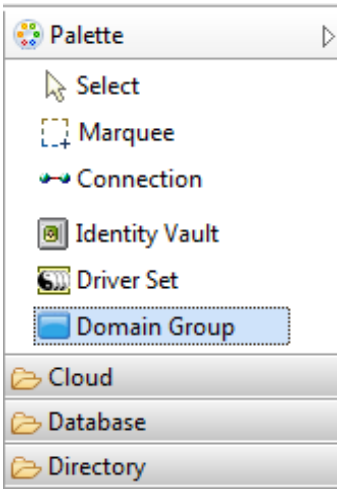
The Architect and Dataflow views contain the same drop-down menu with the same functionality.

Organizing by Domain Groups

- "About Domain Groups" on page 44
- "Key Features" on page 45
- "Creating a Domain Group" on page 45
- "Minimizing (Collapsing) Domain Groups" on page 47
- "Restoring Domain Groups" on page 48
- "Maximizing Domain Groups" on page 48
- "Using a List View of Domain Groups" on page 48
- "Auto-Placement of Neighbors" on page 49
- "Grouping into a New Domain Group" on page 49
- "Ungrouping a Domain Group" on page 51
- "Clearing Contents" on page 51
- "Changing a Domain Group Icon" on page 51
- "Keyboard Support for Domain Groups" on page 52

About Domain Groups

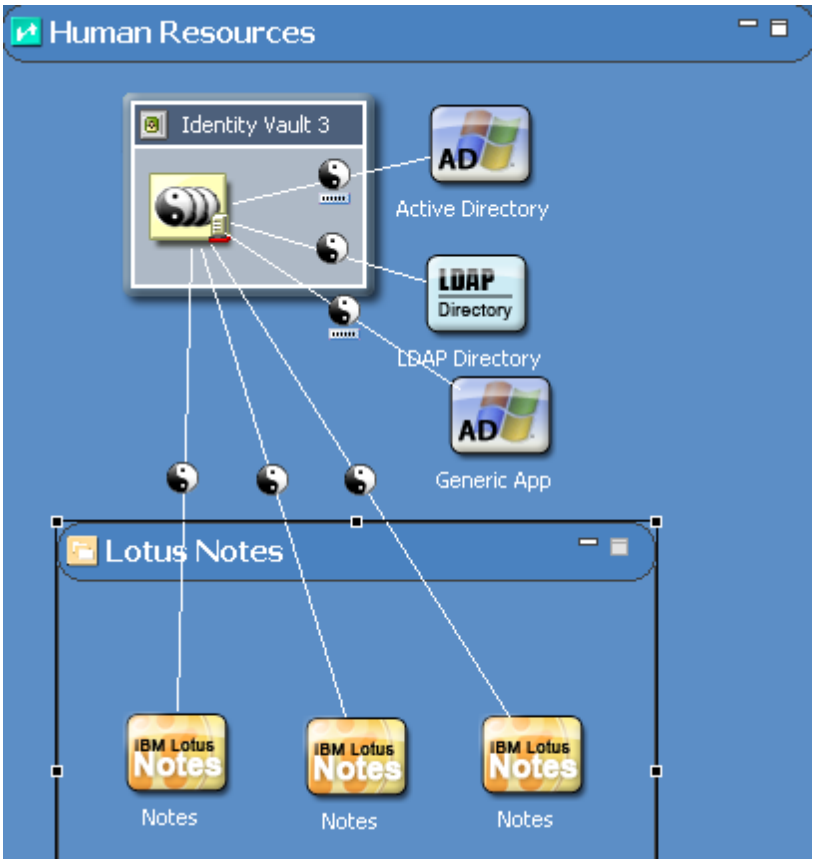
Figure 2-12 *The Domain Group Option on the Palette*



Domain Groups enable you to organize your model into logical groupings that help to keep your diagram clean. Domain Groups have no technical function, and they have no impact on how items and relationships are stored in the Identity Vault. This option is just a tool to help you better organize and view items in the Modeler.

Using Domain Groups is the key to modeling your entire enterprise, no matter how large it is. You can create a model that is manageable, useful, and logical, according to how you want to organize and diagram your enterprise.

Figure 2-13 *A Domain Group in the Modeler*



Key Features

- Change a group name through the Properties view.
- Drag and drop items in and out of groups.
- Minimize or restore groups.
- Move everything in a group.
- Remove everything in a group.
- Nest groups within groups (no limit).
- Resize groups. A minimum size is enforced.
- Ungroup. Remove the group but leave the children.

Creating a Domain Group

1. Drag and drop a Domain Group from the palette to the Modeler space.
2. Organize items inside Domain Group items.

To add another Domain Group, drag and drop one from the palette.

To add an Identity Vault, do one of the following:

- Drag an Identity Vault from the palette.
- Right-click in the Domain Group, then select **New** > **Add Identity Vault**.

  The Add Server to Identity Vault dialog box appears. If you select **Specify a Server**, Designer provides a dialog box that enables you to select an OpenText eDirectory server or specify a server manually.

To add a driver set:

1. Right-click inside an Identity Vault.
2. Select **Add Driver Set**.

To add an application:

1. Right-click a Driver Set object.
2. Select **Add Connected Application**.

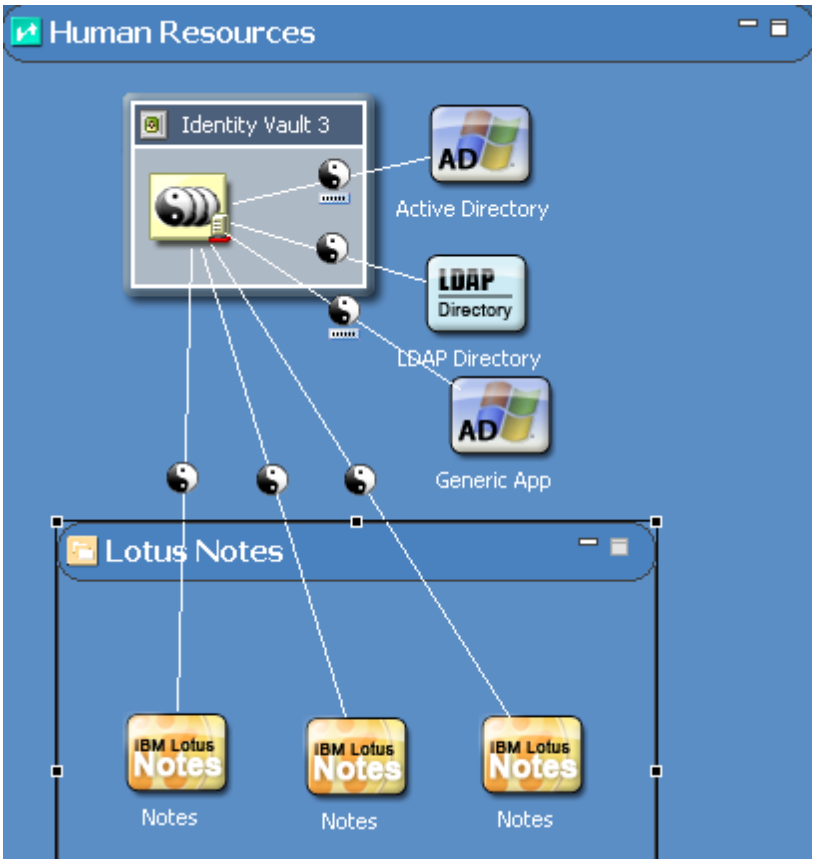
   The application is added to the right of the right-most connected application. If this is the first application, it is placed under the driver set.

The application defaults to a generic application type. To change the type:

1. Right-click the application, then select **Properties**.
2. Select a different application, then click **OK**.

When you add selected items to a Domain Group, the Domain Group expands.

Figure 2-14 *A Domain Group*



If you move an item to the edge of the Domain Group, the boundaries expand, so that the items remain inside the Domain Group. You can drag an item from the Domain Group to remove it from the group.

You can have nested domains. If you expand a nested domain, the outer (hosting) domain automatically increases in size. You aren't required to manually resize parent domains. By expanding, the hosting domain displays the nested domain, so that the nested domain isn't cut off.

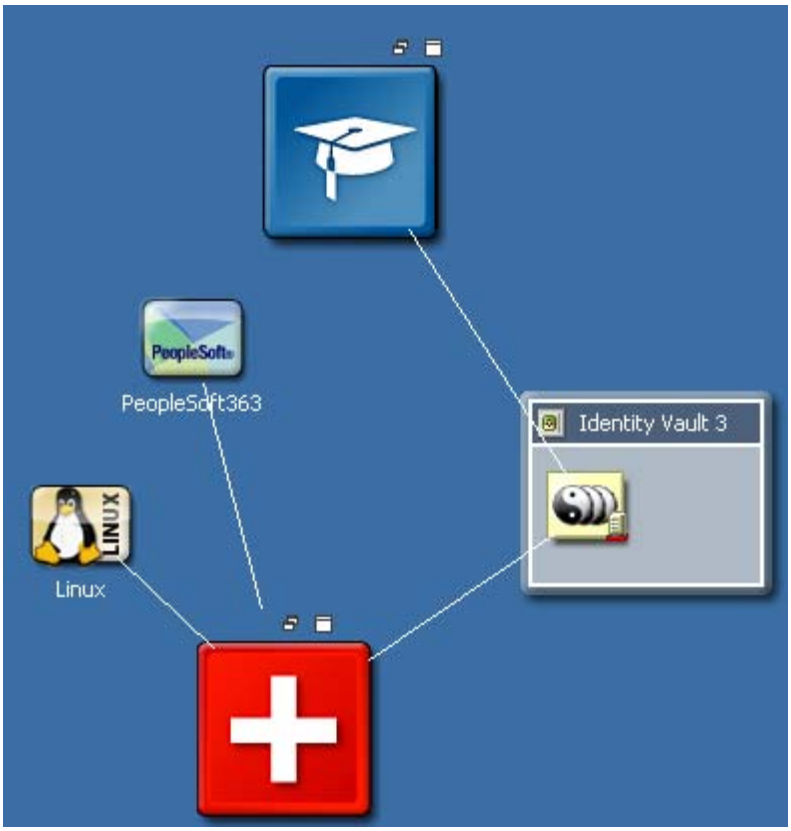
Minimizing (Collapsing) Domain Groups

To minimize a Domain Group, click the **Minimize** icon. When a Domain Group is minimized, it defaults to a random icon. You can use Properties to change the icon. (See "Changing a Domain Group Icon" on page 51.) The icon and minimized state of the group are saved in the Project file.

When a group is minimized, you can't see its contents, nor can you drag new items into the group. However, you can move, rename, or delete it.

When you minimize a group, lines that were connected to items in the group now connect to the group. This functionality enables you to see that there is a relationship with items in the group and items outside the group. Depending on your objects, their relationships, and state of other related groups, multiple lines might collapse into one line.

Figure 2-15 *A Collapsed Group*



When you expand the group, the lines are moved back to the actual items they connect with. This functionality works for any level of nesting of groups.

Restoring Domain Groups

To restore the Domain Group to its original size, click the **Restore** icon.

Maximizing Domain Groups

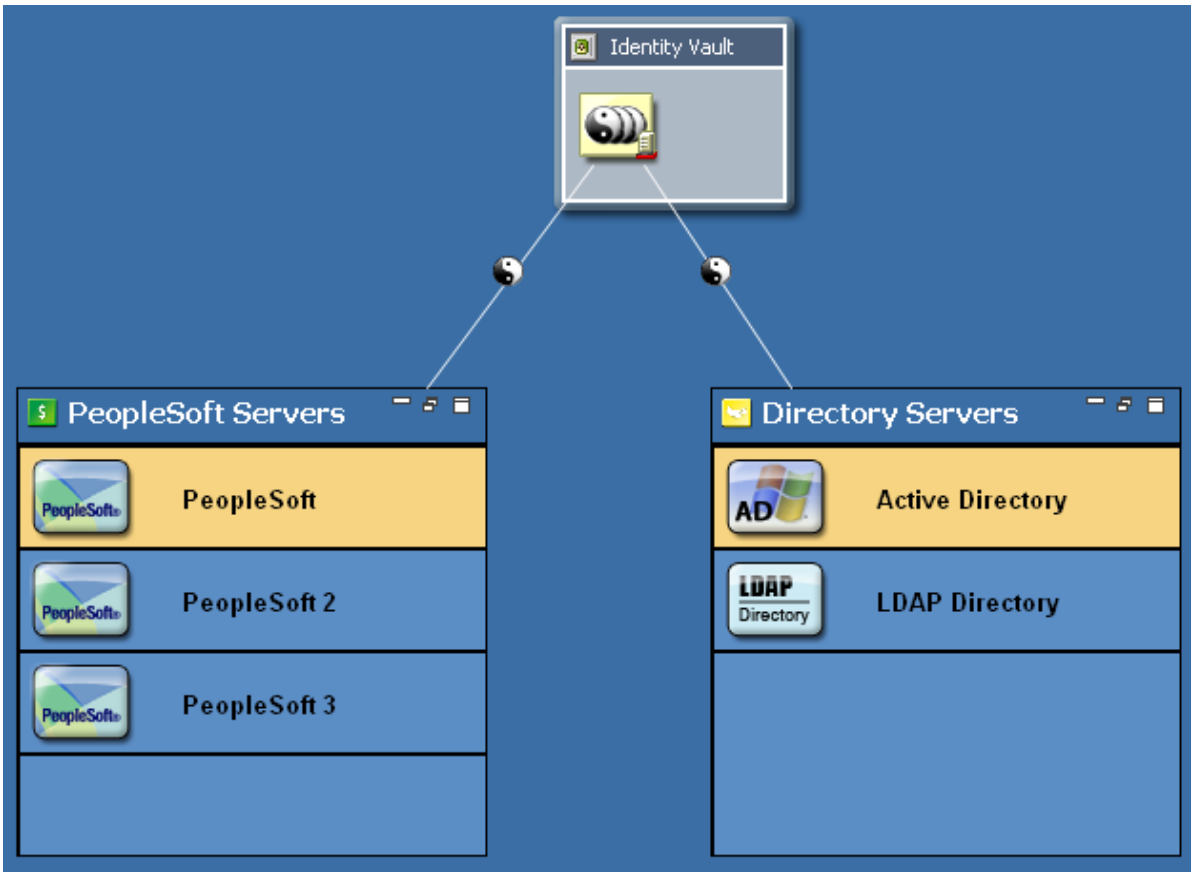
To maximize a Domain Group, click the **Maximize** icon. The group expands to a much larger size. To return it to the original size, click the **Restore** icon.

You can maximize only first-level groups. For inner groups, the Maximize function is disabled.

Using a List View of Domain Groups

To open a Domain Group in a list view, click the **List View** icon. The group lists the applications in a list format. To return it to the original size, click the **Restore** icon.

Figure 2-16 *List View of a Domain Group*



List view of Domain Groups shows only connections of the selected application while the connections of other applications are hidden. You cannot add or delete additional applications in the list view. To perform any operation, right-click the corresponding driver connector.

List view of Domain Groups does not support nesting of Domain Groups or Identity Vaults within a Domain Group. Attempting nesting of Domain Groups or Identity Vaults results in a warning message.

Auto-Placement of Neighbors

To push or pull the neighboring items when you expand or contract Domain Groups, hold down the Ctrl key while you expand or contract the Domain Group. Any item that is to the right or below a Domain Group is affected.

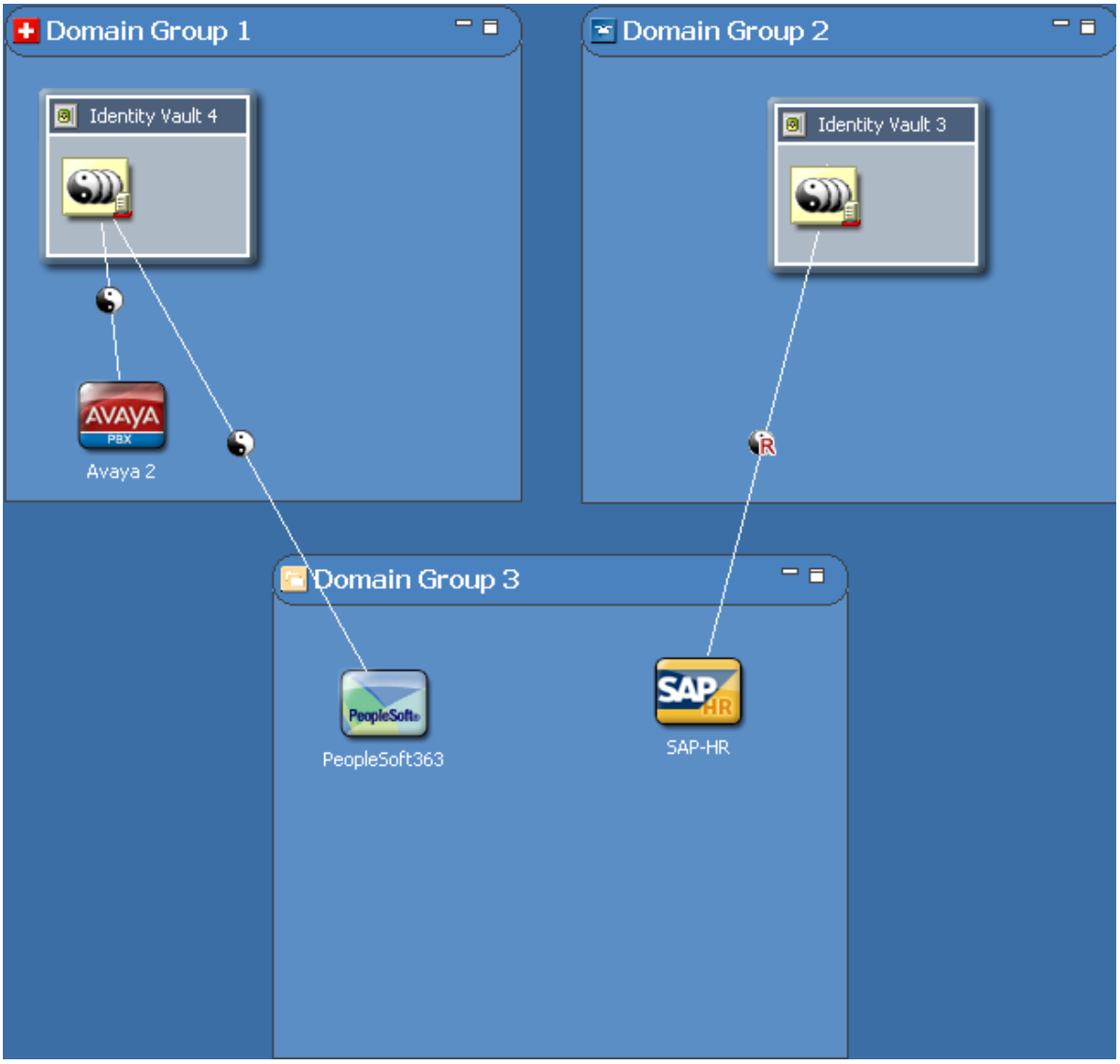
Grouping into a New Domain Group

1. In the Modeler, select multiple items.
2. Right-click, then select **Add to Group**.

The Modeler creates a new Domain Group and adds those items, preserving their relative spacing to each other. This process removes the items from wherever they previously existed and places them in the proper area in the new group.

The following figure illustrates two Applications that have been added to a new Domain Group and removed from their previous groups.

Figure 2-17 *Grouping into a New Domain Group*



Ungrouping a Domain Group

Figure 2-18 *Ungrouping a Domain Group*



To ungroup a Domain Group, right-click it, then select **Ungroup**.

This process removes the Domain Group but leaves all contents where they are, so that they won't be deleted. This is just a way to ungroup the items. Depending on what level you are in the Modeler, the ungrouped items are automatically added to the host group or to the main Modeling space.

Clearing Contents

To remove all contents from the Modeler, click **Model**, then select **Clear All Items**.

To remove all contents from a Domain Group, right-click, then select **Clear Domain Contents**.

Designer prompts you before clearing the Modeler space.

Changing a Domain Group Icon

1. Right-click a Group Domain item in the Modeler, then select **Properties**.
2. Browse to and select an image (for example, `finance.png`).

   Icons for Domain Group components reside in the `Group` directory in the Modeler plug-in directory. By default, Designer opens the `Group` directory.

Designer supports .GIF, .JPEG, .PNG, and Windows .BMP formats. You can add your own icons to the Group directory.

3 Click **Open**, then click **OK**.

The minimized 16x16 version of the image also now appears in the Domain Group title bar.

As you add Domain Group items, Designer randomly assigns icons from the Group directory to the new Domain Group.

Keyboard Support for Domain Groups

Table 2-3 *Shortcut Keys for Domain Groups*

Keystroke	Description
Alt+Down-arrow	Navigates into a Domain Group
Alt+Up-arrow	Navigates out of a Domain Group
Delete	Deletes the selected items

Connecting Applications

- "Automatic Connections" on page 52
- "Connection Target Highlights" on page 53
- "Automatically Creating Objects" on page 53
- "Auto Redraw" on page 54
- "Manually Connecting" on page 54
- "eDir-to-eDir Connections" on page 54
- "Multiple Driver Connections" on page 55
- "Straightening Connections" on page 56
- "Reconnecting" on page 57
- "Driver Icons" on page 57
- "Selected Drivers" on page 58
- "Auto-Layout of Imported Objects" on page 58
- "Keyboard Support for Connections" on page 58

Automatic Connections

When you drag an application into the Modeler space, and the Modeler contains a driver set, Designer automatically draws a connecting line between the Driver Set object and the application.

When you use the palette's Connection function to connect an application to an Identity Vault, you can begin or end your driver line at the Identity Vault. The line automatically connects to a driver set in an Identity Vault.

If the Identity Vault contains more than one driver set, the Connection function connects the driver line to the first driver set. This functionality also works for multi-driver connections.

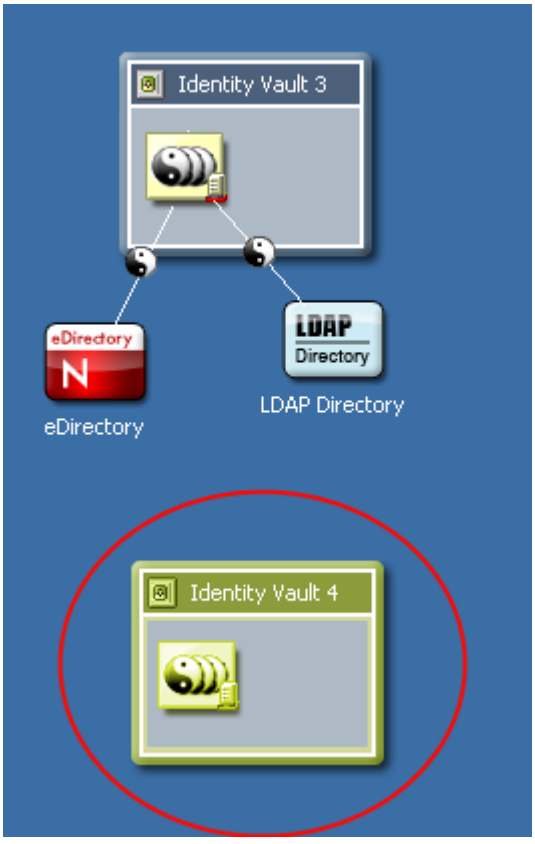
All multi-driver driver lines are bendable. You can lay them out so that the lines don't overlap at any angle. Also, you can reconnect multi-driver connections.

If an Identity Vault has multi-driver connections in a Domain Group and you minimize that Domain Group, a single collapsed line represents all of the multi-driver connections.

Connection Target Highlights

When you drag an application across the Modeler space, the closest Identity Vault and closest driver set in that Identity Vault are highlighted. The highlights indicate the item that the application will connect with when you drop the application.

Figure 2-19 *Connected Objects*



Automatically Creating Objects

If you drop an application into the Modeler space, and that space has no Identity Vaults, Designer automatically creates an Identity Vault.

If you add a driver application in the Modeler by right-clicking in the Modeler, then selecting **New** > **Application**, the driver application is now added at the place where you right-clicked. This makes it easier to locate items in the view.

Auto Redraw

If you move items, lines are automatically redrawn.

Manually Connecting

To manually connect an application to a driver set:

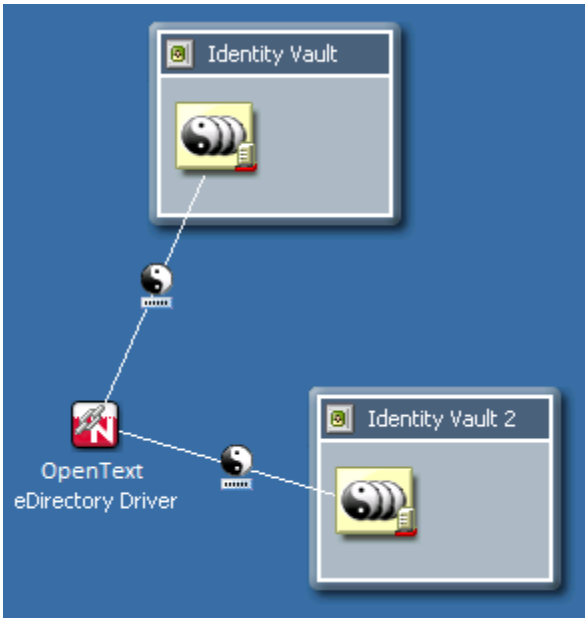
1. Click **Connection** in the palette.
2. Draw a line between the application and the driver set.

To reconnect an application, select the driver line, then drag one end of the line to another driver set or application.

The drag gesture gravitates the line towards the nearest connectable point. This functionality helps you know what you can connect to and where you can connect the item. If you try to connect to something that isn't allowed, the cursor usually indicates so, or nothing happens when you drop the item.

eDir-to-eDir Connections

Figure 2-20 *eDir-to-eDir Connections*



An eDir-to-eDir connection is a special type of connection. It is used frequently in OpenText Identity Manager environments. This connection is a way to configure two OpenText eDirectory drivers to communicate directly with each other. (No other drivers are able to communicate directly with any other type of driver.) This type of connection is most commonly used for synchronizing a local directory tree with a Identity Vault.

To create an eDir-to-eDir connection, do one of the following:

- Drag a line between two Identity Vaults
- Drag a line between two driver sets

When you connect a line between two OpenText eDirectory applications, the line automatically turns into an eDir-to-eDir connection. See the illustration in "Viewing an eDir-to-eDir Driver" on page 256.

To disconnect an eDir-to-eDir connection, right-click an eDir item, then select **Disconnect eDir-to-eDir**. Designer creates two new OpenText eDirectory applications and redirects each driver to its respective application. A new driver is not created. No data is lost. Designer keeps the same drivers.

If you delete one side of an eDir-to-eDir connection, Designer converts the remaining half into a regular driver connection to an OpenText eDirectory application.

Multiple Driver Connections

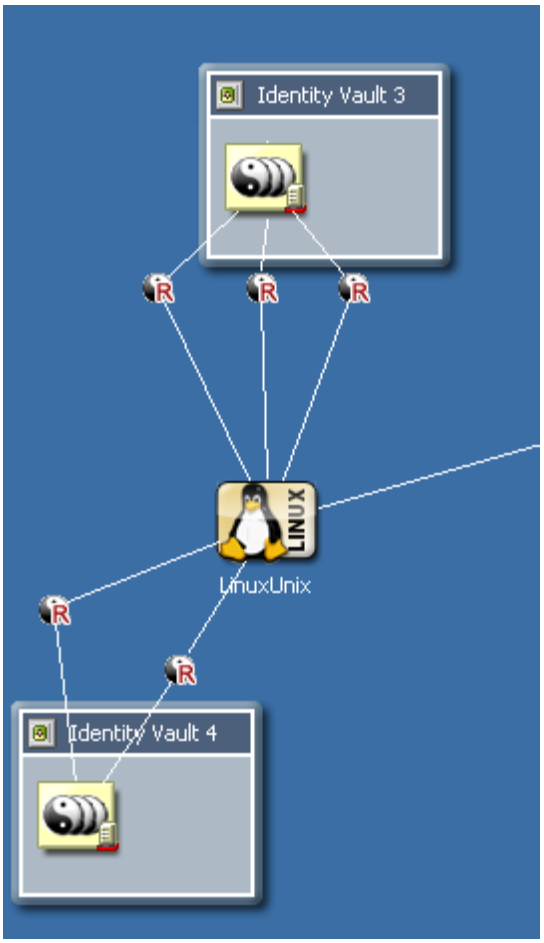
To connect more than one driver from a driver set to an application:

1. Select **Connection** in the palette.
2. Connect the driver set and the application again and again.

Each time you connect, a new line is added. All lines are bendable, so that the lines don't overlap. To get the model to look optimal, you probably need to move the application slightly from its default position.

You can also connect more than one driver to a single application. This actually causes the application to act as a hub. Each driver can connect to and authenticate to the application or system the same or differently, depending on your needs. Each driver can access the same part of the application or system or different parts (for example, different tables in a database). The Modeler lets you diagram a layout according to your needs.

Figure 2-21 *Multi-Driver Connections*



Straightening Connections

To straighten connecting lines:

1. Press Ctrl, then select one or more items in the Modeler.
2. Right-click, then select **Straighten Connections**.

What is straightened depends on what you select:

Table 2-4 Straightened Connections

Selected Item	What Is Straightened
A driver	That driver's line
An application	The connecting driver's line
An Identity Vault	All lines that originate from that driver set in that Identity Vault
A Domain Group	Everything in the Domain Group
A project (selected by clicking the Modeler's background)	Everything in that project

Lines are straightened only if they are less than 20 pixels from a north, west, south, or east alignment. The intent of this operation is to quickly nudge lines that are almost straight, so that they become perfectly straight.

This nudging removes the tedium of meticulously dragging items into perfect alignment and being concerned with the pixels. If a line isn't almost straight, it is left alone. In fact, the Straighten Connection operation is disabled unless the selected items qualify to be straightened. If some of the selected items qualify but others don't, the operation is still enabled, but only eligible lines are straightened.

Reconnecting

To reconnect components, do one of the following:

- Drag the end of a line (driver) from one application to another.
- Drag the end of a line (driver) from one driver set to another.

Driver Icons

Table 2-5 Driver Icons

Icon	Description
	A driver. The entire line between a driver set object and an application represents a driver.
	A remote driver.
	A firewall. Indicates that the driver is communicating across a firewall.

To see, turn on, or turn off driver icons:

1. Right-click a driver line.
2. Select an option (for example, **Mark as Firewall**) to turn on or turn off.

Selected Drivers

As you move the mouse over a driver, the line thickens so that it is more obvious. You can click and interact with this line.

Auto-Layout of Imported Objects

When you import objects from the directory, they are automatically laid out, connected with lines, and assigned an icon that matches objects and relationships as closely as possible.

For example, if you import a Driver Set object, Designer imports all of the drivers and connects them with lines. Also, each driver points to an application icon. Application icons include the following:

- The exact Application icon (for example, Avaya or PeopleSoft)
- The image stored on the driver

  The image is embedded in a square application icon.
- A generic application icon

  If no image is stored on the driver, Designer supplies an icon for one of the following applications:

  - Generic
  - JDBC
  - LDAP
  - Delimited Text

  The auto-layout mechanism uses the layout topology that you have selected. The default is Fan Out - Bottom. You can customize this setting in Preferences. See “Modeler” on page 505.

Keyboard Support for Connections

Table 2-6 *Shortcut Keys for Connections*

Keystroke	Description
/	Navigates to the item’s next connection
\	Navigates to the item’s previous connection

Aligning and Laying Out Components

- “Alignment Hints” on page 59
- “Using Rulers” on page 60
- “Using a Grid” on page 62
- “Distributing Applications” on page 63
- “Auto-Layouts” on page 63
- “Layouts to Use for Imports” on page 64

Alignments place objects in the same horizontal or vertical plane. Alignments help you see relationships in your model. You can align or attach items to the left, center, or right of alignment guides.

When you move the guide, attached items move with it, staying attached in the same relative positions.

To align components:

1. Press Ctrl, then select more than one item.
2. Right-click, then select **Align**.
3. Select an alignment option.

You can also attach an item by dragging it to a guide. After you wait a moment, the guide line is highlighted, indicating that the item is attached.You can align within the same group but not across groups.

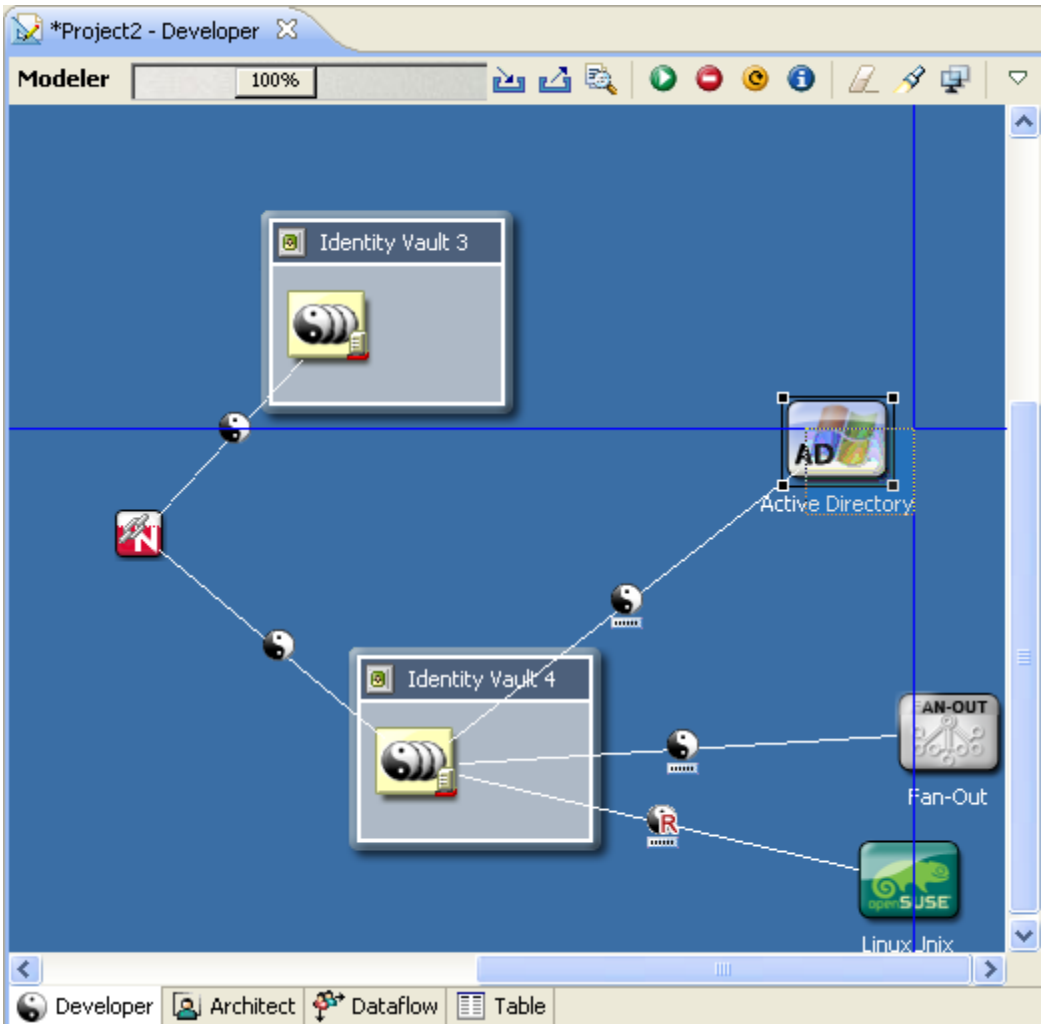
Guides that you set up are restored the next time that you run Designer. You don't need to re-create them.

Also, the alignments and attachments (left, center, or right) are stored in the project on a per-item basis, so that they are also restored.

Alignment Hints

Click **View** > **Alignment Hints** to automatically show horizontal and vertical "hint" lines as you drag items into vertical or horizontal alignment with neighboring items.

Figure 2-22 *Alignment Hints*

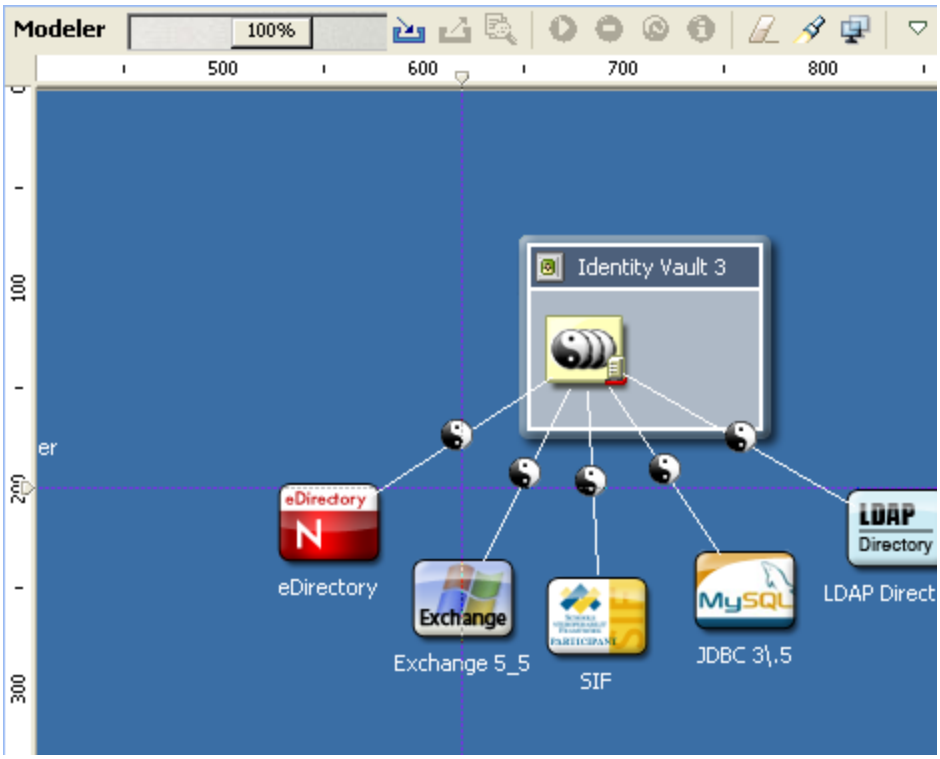


The Alignment Hints feature is off by default. To turn it on, click **View** > **Alignment Hints**.

Using Rulers

To turn on the horizontal and vertical rulers:

1. Click the Modeler space to make it active.
2. Click **View** > **Rulers**.



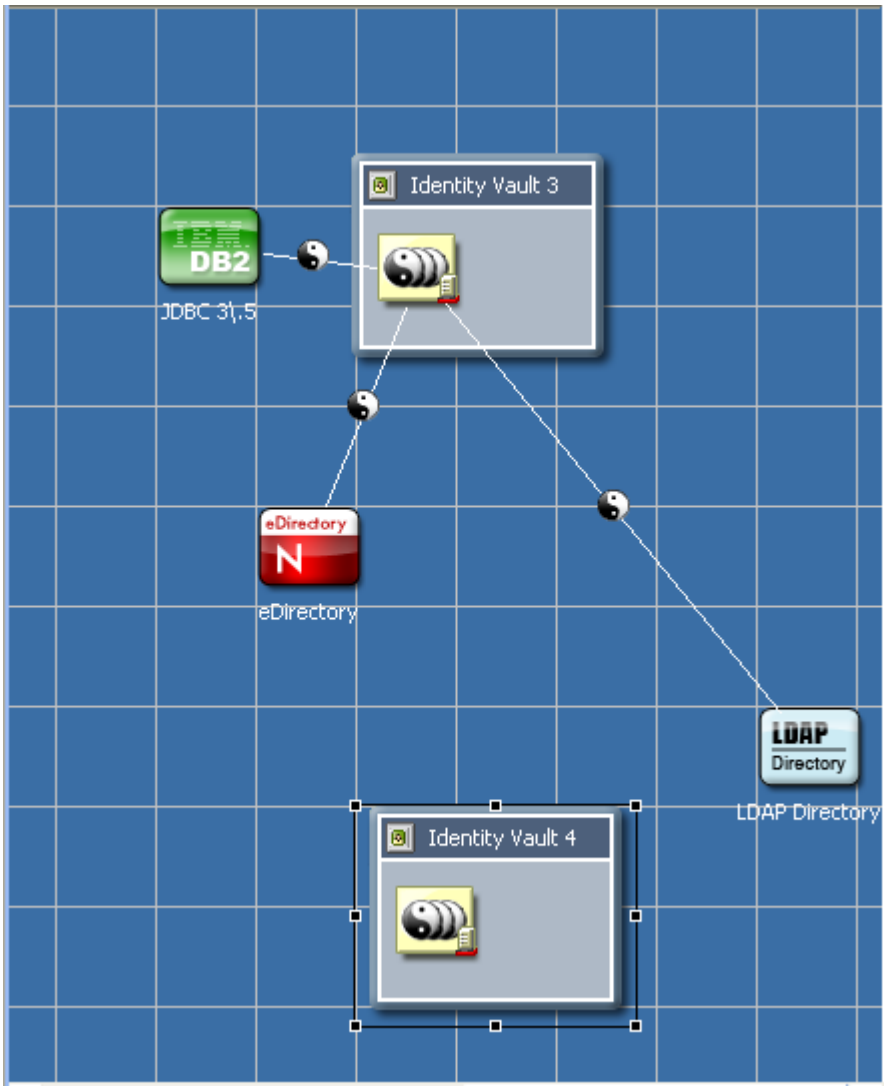
To create a guide (line), click either ruler.

To anchor items to a guide, drag the items in the model to the line.

To simultaneously move all anchored items, drag the line.

Using a Grid

Figure 2-23 *The Modeler's Grid*



When the grid is on, the snap-to-grid functionality is on.

To turn grid lines on and off:

1. Click the Modeler, so that the Modeler is the active view.
2. Click **View** > **Grid**.

To coerce objects to not align with the grid, temporarily turn off snap-to-grid by holding down the Alt key. (Linux doesn't support this functionality.)

To constrain items to north-south or east-west coordinates, press Shift while dragging the items.

To change the grid size:

1. Click **Window** > **Preferences** > **NetIQ** > **Identity Manager** > **Modeler** > **Display**.
2. Type a value in the **Grid Width** field.

Distributing Applications

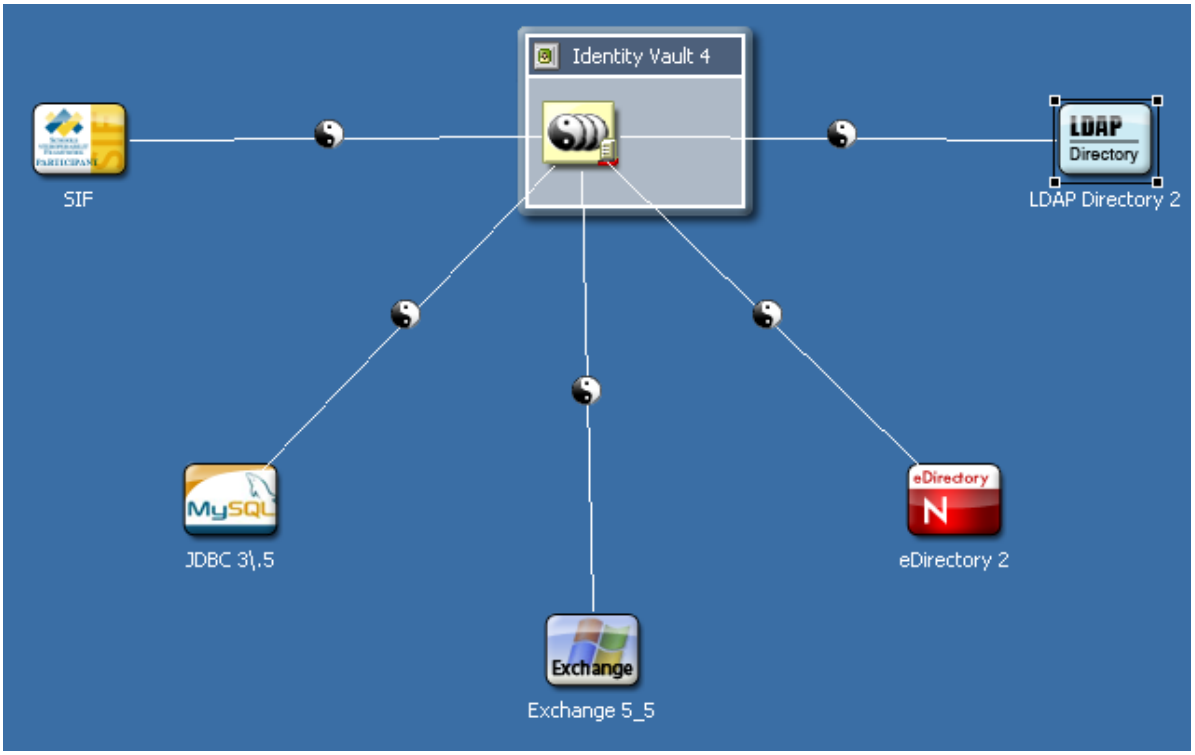
To equally distribute (space) applications horizontally or vertically:

1. Press Ctrl, then select three or more items.
2. Right-click, then select **Distribute**.
3. Select a distribution (for example, **Vertical**).

Auto-Layouts

Designer ships with a number of predefined layout topologies: circle, half-circle, star, box, and different fan-out layouts.

Figure 2-24 *A Half-Circle Layout*



These layouts are set on a per-driver-set basis. Therefore, each driver set can have its own layout.

To select a layout:

1. Right-click a driver set, then select **Arrange Applications**.
2. Select an arrangement (for example, **Fan Out - Left**).

   If your model has an incorrect layout, the layout options are dimmed.

After you set a layout, applications that you connect will automatically snap into that layout. Certain connected objects (for example, multi-driver connections, eDir-to-eDir connections, and applications that are connected but reside in a different Domain Group) are ignored. They aren't included in the layout, and they don't disturb it.

An option on the Arrange Applications submenu on the Modeler's context menu enables you to expand or contract the layout arrangement. This option makes all spokes of the layout longer or shorter when you drag a slider.

Layouts to Use for Imports

To specify what layout to use on new driver sets that you import:

1. Select **Window** > **Preferences** > **NetIQ** > **Identity Manager**.
2. Click **Modeler** > **Layouts**.
3. Select an arrangement (for example, **Half Circle**), then click **OK**.

Editing Multiple Objects

You can open multiple objects and edit them at the same time. These objects must be of the same type (for example, policies).

To find out whether you can edit an object, right-click it. If **Edit** displays among the menu items, you can edit that object.

1. In the Outline view, expand the project that contains the objects that you want to edit.
2. Select the objects.
3. Right-click, then select **Edit**.
4. Edit the objects.

   You can copy and paste from one editor to another. Data must be of the same type.

Modeling Active Directory Domain Controllers

- "Configuring a Connection" on page 64
- "Discovering Controllers" on page 65
- "Information about Domain Controllers" on page 65

Configuring a Connection

You can configure an LDAP connection to an Active Directory system so that you can discover its domain controllers.

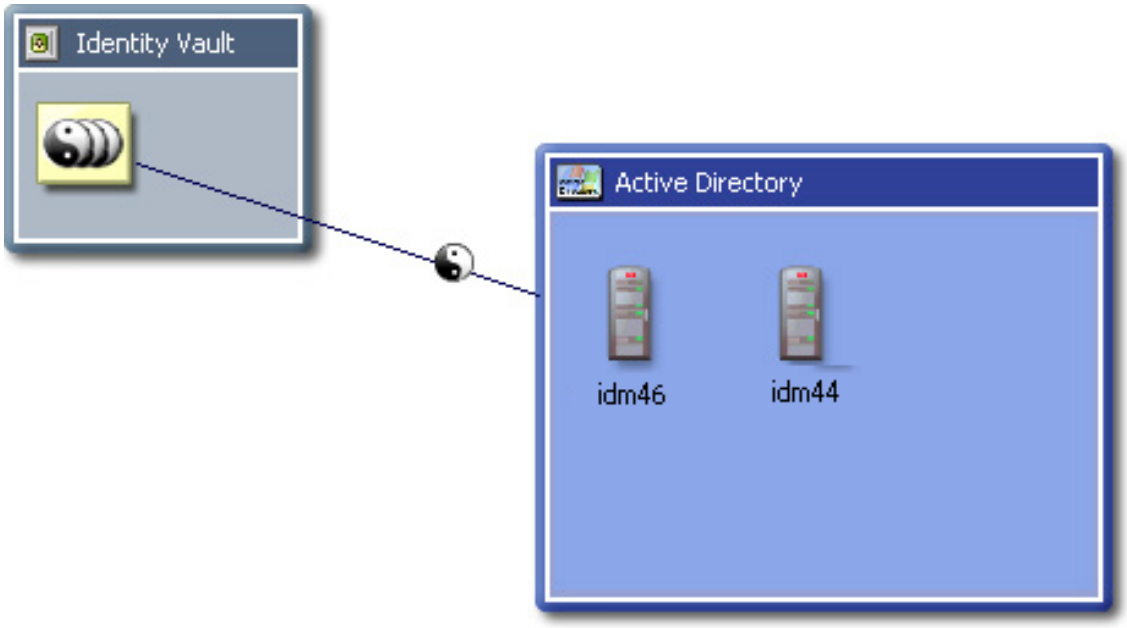
1. Right-click the Active Directory application, then select **Properties** > **Connectivity**.
2. Complete the LDAP authentication information.

   As you tab from the **Host** field to the **User** field, Designer automatically builds a full user context. You can modify this context.

Discovering Controllers

1 Right-click the Active Directory application.

2 Select **Discover Domain Controllers**.



If DesignerDesigner finds any controllers, it lays them out and expands the Active Directory application as a container.

Information about Domain Controllers

Information about each controller is loaded into the Modeler. To view this information, edit the Domain Controller object and select the AD Domain page.

If the LDAP connection information is filled out, you can reread the information from that system by clicking the **Refresh** icon.

Saving Your Model

To save your model, do one of the following:

- From the main menu, select **File** > **Save** (or **Save All**).
- From the main menu, select **File** > **Close** > **Yes**.
- Click the **X** in the Modeler's tab, then select **Yes**.

For more information, see "The Project View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

3 Configuring Objects in Designer

Designer allows you to easily view, configure, and modify settings for Identity Vaults, driver sets, drivers, and managed systems.

- "Viewing Object Properties" on page 67
- "Configuring a Domain Group" on page 70
- "Configuring Identity Vaults" on page 70
- "Configuring Servers" on page 75
- "Configuring Driver Sets" on page 76
- "Configuring Libraries" on page 83
- "Configuring Drivers" on page 84
- "Configuring Policies" on page 110
- "Configuring Resource Objects" on page 111
- "Configuring Categories" on page 111
- "Configuring Groups" on page 111
- "Configuring Packages" on page 111
- "Configuring Package Content" on page 116
- "Configuring Prompts" on page 116
- "Configuring Global Configuration Objects" on page 118
- "Configuring Jobs" on page 118
- "Configuring ID Policy Containers" on page 119
- "Configuring ID Policies" on page 120
- "Configuring a Notification Template" on page 122
- "Configuring Application Properties" on page 122
- "Adding Prompts to a Driver Configuration File" on page 127
- "Synchronizing Passwords" on page 127

Viewing Object Properties

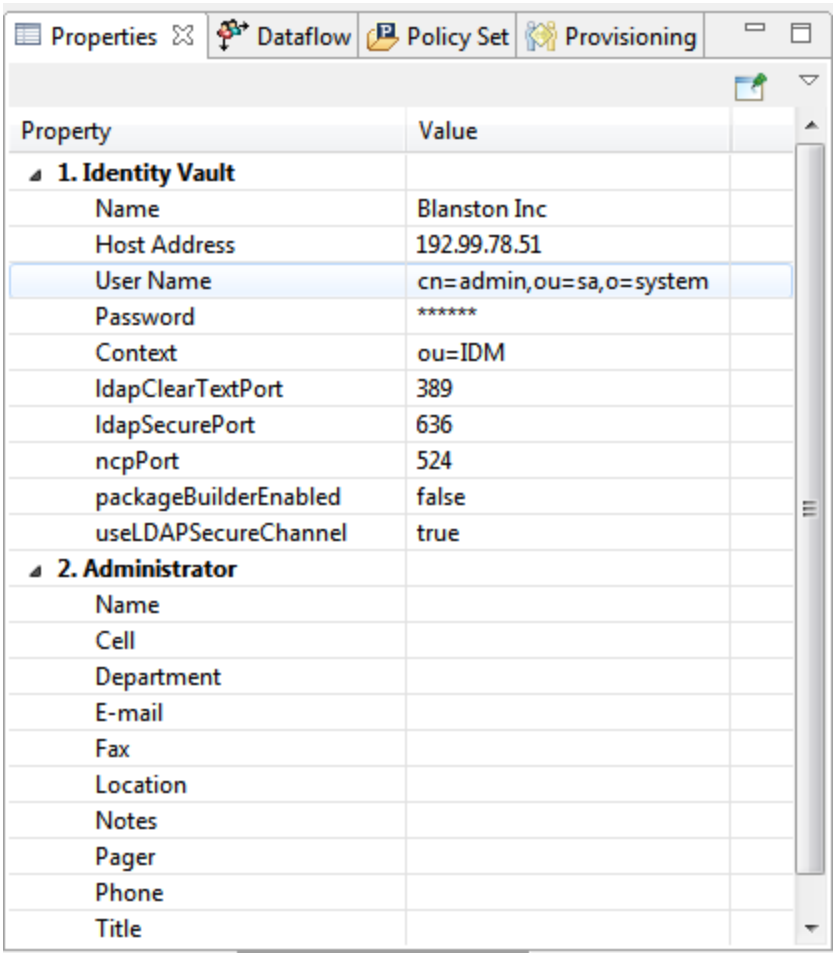
To quickly view or edit properties of items (for example, an Identity Vault or a driver), you can use the Properties view or a Properties dialog box.

- "Properties View" on page 68
- "Properties Dialog Box" on page 68
- "Operations Relating to Properties" on page 70

Properties View

If the Properties view is open when you select an item in the Modeler, information about that item displays in the Properties view. You can then quickly view or edit information. For example, the Properties view of an Identity Vault looks similar to this:

Figure 3-1 *The Properties View of an Identity Vault*



To open the Properties view, click **Window** > **Show View** > **Other > General > Properties**. For additional information, see "The Properties View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

Properties Dialog Box

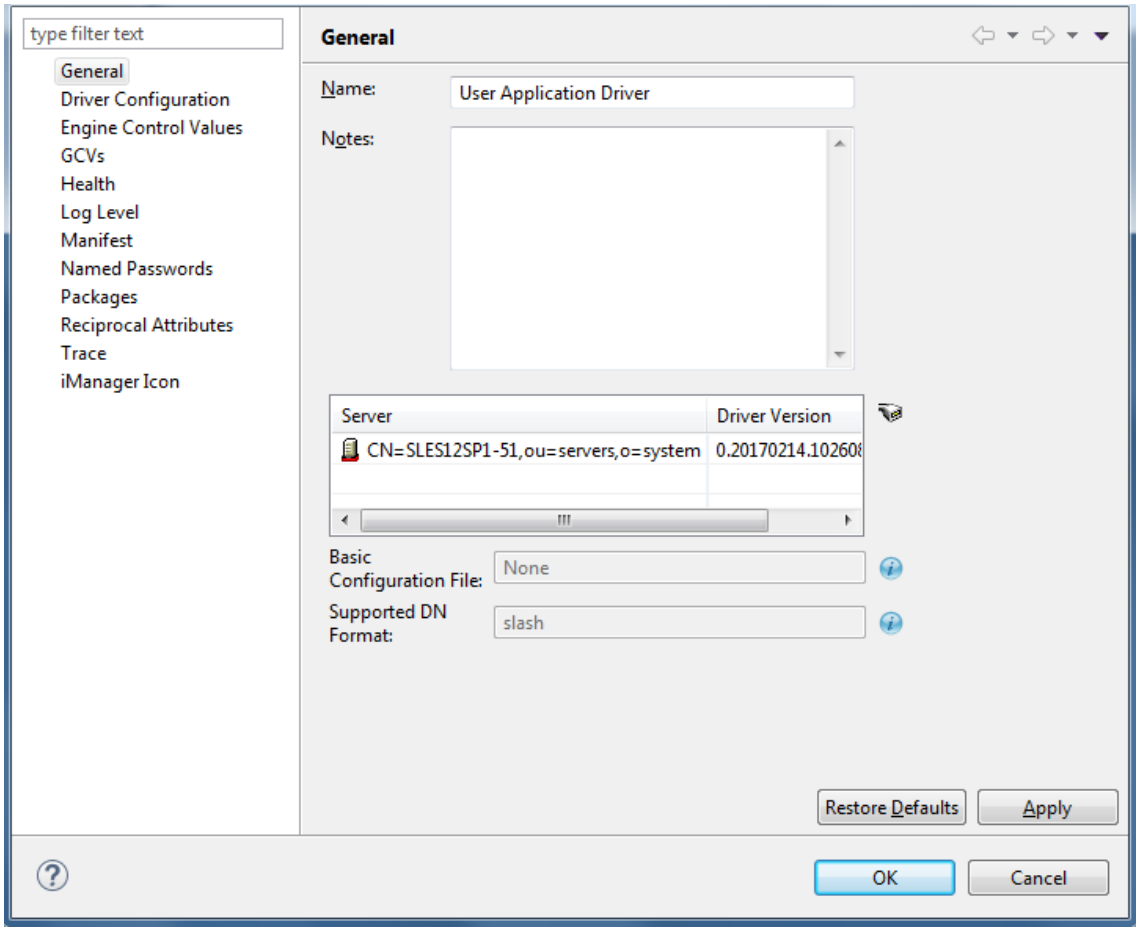
The list of property pages in the Properties dialog box is organized alphabetically across Designer with the exception of the General page, similar to that of Eclipse.

To view or edit properties of items:

1. Open the Properties dialog box by doing one of the following:
   - Double-click an item in the Modeler or in the Outline view.

- Right-click an item (for example, an Identity Vault) in the Modeler or Outline view, then select **Properties**.
- Select an item, then press Enter.
- Select an item, then select **File** > **Properties**.
- Select an item, then select **Model** > [object] > **Properties**.

The following figure illustrates a driver's properties page:



2 Edit settings, then click **OK** to save.

Operations Relating to Properties

Table 3-1 *Operations Relating to Properties*

Operation	Description
Open the Properties view	Click **Window** > **Show View** > **Other** > **General** > **Properties**.
Open the Properties dialog box	Double-click an item, or right-click the item, then select **Properties**.
Edit settings	You can edit the settings of any item selected in the Modeler or Outline view.
View a server's properties	In the Outline view, right-click the server icon, then select **Properties**.
Save to memory or disk	When you click **Apply** or **OK** in a properties dialog box, changes are committed to memory. However, changes are not saved to disk unless you select **File** > **Save**.

Configuring a Domain Group

To view or change a domain group's settings, double-click the domain group.

1. To change the domain group's icon, click **Browse**, then navigate to and select an image file.

   By default, the **Browse** button opens the `icons/group` folder in the `com.novell.designer.core` plug-in. The default image selected is `administrative.png`. To select a different image, double-click the new image.
2. Click **Apply**.
3. To change the name of the domain group, edit the **Name** field.
4. Add details in the **Notes** pane.
5. Click **OK**.

   The image (for example, `administrative.png`) appears to the left of the domain name in the Modeler.

Configuring Identity Vaults

To view or change an Identity Vault's settings, double-click the Identity Vault object in the Outline view or the Modeler.

The Identity Vault Properties page has several options. In addition, you can configure a hostname in the `hosts` file.

- "Configuration" on page 71
- "Administrator" on page 73
- "Workflow Forms" on page 73

- “Packages” on page 74
- “Server List” on page 75
- “iManager” on page 75
- “Local Hostname” on page 75

Configuration

The following table contains a description of each of the Identity Vault configuration settings.

Table 3-2 *Configuration Settings for an Identity Vault*

Field	Description
Vault name	The name of the Identity Vault object. The default is Identity Vault.
Host	The OpenText eDirectory host where you plan to log in and deploy.
Username	The eDirectory username in LDAP format that has sufficient rights to make changes to objects associated with this deployment. For example, `cn=admin,ou=sa,o=system`.
Password	The password for the eDirectory username.
Save Password	Saves the password permanently, so you are authenticated into this Identity Vault each time you open Designer. If you use this option, the password is saved locally in Designer’s file system and is not secure. If you do not select this option, the password is remembered only until you close Designer.

Field	Description
Secure Connection	You can enable a secure or a non-secure connection between Designer and the Identity Vault. **Secure Connection:** In a secure connection, the LDAP server listens on port 636 by default. If you configured the secure port as 700, specify this port number in the **Host** field. For example, `192.99.78.51:700`. When connecting through a secure port, Designer prompts you to import the Identity Vault's Certificate Authority certificate into Designer. You must accept this certificate to establish a secure connection with the Identity Vault. Designer provides the following options to accept the certificate: ◆ **Accept this certificate permanently**: Instructs Designer to not prompt you again for accepting the certificate for future authentication with an Identity Vault. When this option is enabled, Designer permanently stores the certificate in `/opt/netiq/tools/Designer/configuration` or `C:\netiq\idm\apps\Designer\configuration` directory. ◆ **Accept this certificate temporarily for this session**: Instructs Designer to stop prompting you for accepting the certificate until the connection expires. ◆ **Do not accept this certificate and do not authenticate**: Instructs Designer to reject the certificate. The connection is not established without a certificate. ◆ **Remember this selection of certificate import**: Instructs Designer to remember your choice of certificate import for future authentication. To change the certificate import settings, go to the Modeler preferences and click **Window > Preferences > NetIQ > Designer > LDAP Connection**. **Non-secure Connection:** If you use a non-secure connection, all the information you enter, such as user names and passwords, is sent over the wire in clear text. The LDAP server listens on port 389 in a non-secure connection by default. **NOTE:** For security reasons, named passwords will be deployed only in a secure connection. If you want to use a non-default port, specify the port number of the LDAP server for establishing a connection. To change the secure and non-secure port numbers, open the Properties view of an Identity Vault and change the default values for **ldapSecureTextPort** and **ldapClearTextPort** fields respectively. If you add additional servers to the Identity Vault after initial connection, Identity Vault automatically assigns the default secure and non-secure port numbers to the server.

Field	Description
Test Connection	Selecting this button allows the user to create, or, if a connection is unresponsive, to re-create a connection to the Identity Vault. If a connection has not been established to the Identity Vault, the button displays **Test connection**. After a connection is established, the button displays **Refresh connection**.
Deploy Context	The default DN container assigned to all driver sets that are associated with this Identity Vault. If you specify a DN container on the Driver Set object, that setting takes precedence over the default setting.
Enable Package Developer Mode	Enables additional features in Designer to allow developers to create packages. For more information, see Section 7, "Developing Packages," on page 193.

Administrator

The **Administrator** option is divided into three sections. Entering information in these sections is optional.

- **Personal Information:** Lets you enter information specific to the Identity Vault, such as Name, Title, Department, and Location.
- **Contact Information:** Lets you enter information such as Email, Phone, Cell Phone, Pager, and Fax.
- **Notes:** Allows you to type any reminders you might need for future reference.

Workflow Forms

The **Workflow Forms** option allows you to configure the Form Backend URL for each Identity Vault instances individually.

To configure a URL, click **Add Registry** and then specify the DNS or IP Address of the server where Workflow Forms will be rendered. Similarly, to delete an existing URL, click **Delete Registry**.

NOTE: If you do not configure the Form Backend URL using this option, the configuration details specified in the `ServiceRegistry.json` file will be considered. The `ServiceRegistry.json` file can be located at the following directories based on your platform:

- **Linux:** `/<designer installed location>/configuration`
- **Windows:** `C:\netiq\idm\apps\Designer\configuration`

Packages

The **Packages** option allows you to manage any packages at the Identity Vault level. A package at the Identity Vault level contains Notification Templates or sample data such as users or the Identity Vault structure. Identity Vault packages are applied to all of the drivers that reside in the selected Identity Vault.

The following table lists the options available to manage packages. For more information about packages, see Chapter 6, “Understanding Packages,” on page 171.

Table 3-3 *Managing Packages Options*

Options	Descriptions
Add package	Adds a package to the Identity Vault. You must add a package before you can install a package. Click the **Add package** icon, then select the package to install and click **OK**.
Create package	The **Create package** option is only available if the **Enable Package Developer Mode** is selected in the Identity Vault Configuration page. Only developers create packages for redistribution.
Package	Lists the name and current state of the package.
Version	Lists the version of the package.
Upgrades	Indicates that there is a newer version of a package imported into the package catalog, but it has not been installed. The package needs to be upgraded.
Operation	Lists the following operations that can be performed on a package: • **Install:** The **Install** option is only available after a package is added to the Identity Vault. Select **Install**, then click **Apply** to install the package. • **Uninstall:** The **Uninstall** option is only available after a package is installed to the Identity Vault. Select **Uninstall**, then click **Apply** to uninstall the package. • **Upgrade:** The **Upgrade** option is only available if there is a newer version of the package available for installation. Select **Upgrade**, then click **OK** to upgrade the package. • **Downgrade:** The **Downgrade** option is only available if you have upgraded a package and the older package is installed in the package catalog. Select **Downgrade**, then click **OK** to downgrade the package. • **Revert Customizations:** The **Revert Customizations** option is only available if you have made changes to the policies that are installed with a package. Select **Revert Customization**, then click **Apply** to remove the customization.

Server List

The **Server List** option displays the servers that are associated with the selected Identity Vault. You can add, edit, or remove the server entries.

NOTE: If you select the option to allow a default server to be created, that server shows up as Default Server.default_container in the list. You cannot deploy a driver set into an existing OpenText eDirectory tree if you have Default Server.default_container in the Server List. You must first remove this reference and add a OpenText Identity Manager server in an OpenText eDirectory tree.

---

iManager

The **iManager** option displays the URL that Designer uses to launch the iManager administrative tool. You can modify this URL as needed.

To launch iManager from Designer, select **Tools** > **iManager**.

Local Hostname

If desired, Designer supports designating a hostname for your Identity Vault by adding an entry to the `hosts` file of your local operating system. After assigning a hostname to the Host address of your Identity Vault, you can use the hostname instead of an IP address or DNS name to access the Identity Vault.

For example, if your Identity Vault has a host address of 192.168.100.254, you can associate the name *ID-VAULT* to that address in your local `hosts` file. Then, in Designer, you can refer to the Identity Vault by the name *ID-VAULT* instead of using the IP address.

For more information about using your local hosts file, consult your operating system's documentation.

Configuring Servers

1. Right-click the server icon in the Outline view.
2. Select **Properties**.

Table 3-4 lists settings for the Server Properties page:

Table 3-4 *Settings for the Server Properties Page*

Field	Description
Name	The name of the Server object in LDAP format. For example, `cn=server1,ou=servers,o=system.` The Identity Vault lists the server. You can browse to and select the server.

Field	Description
Context	The server's context in LDAP format. For example, `ou=servers,o=system.` The Identity Vault assigns the context. You can browse to and select the context.
Host address	The server's IP address.
DNS name	The domain name or complete directory context name.
Identity Manager version	The version of OpenText Identity Manager that is running on the server. The default is Identity Manager 4.7. You can change the version by using the drop-down list. See "Changing the OpenText Identity Manager Version" on page 130.
eDirectory version	The version of OpenText eDirectory that the server is using.
Assigned Driver Set	The driver set the server is assigned to.
Notes	Information that you want to specify, to help you maintain the server.

OpeUse the **Contact Information** tab to provide information about the person to contact and other items of interest concerning the server.

NOTE: If you add additional servers to the Identity Vault after initial connection, Identity Vault automatically assigns secure and non-secure port numbers to the server.

Configuring Driver Sets

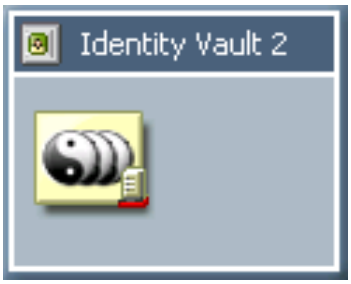
A driver set is a container that holds OpenText Identity Manager drivers. Only one driver set can be active on a server at a time. As a result, all active drivers must be grouped into the same driver set. To view or change settings, double-click a driver set in the Modeler.

- "Driver Set General Options" on page 76
- "Driver Set Configuration" on page 77
- "Driver Set Global Configuration Values" on page 78
- "Java Environment Parameters" on page 78
- "Driver Set Log Levels" on page 79
- "Driver Set Named Passwords" on page 79
- "Driver Set Packages" on page 79
- "Driver Set Server List" on page 80
- "Driver Set Trace" on page 81

Driver Set General Options

When you create an Identity Vault, a driver set is added to the vault by default.

Figure 3-2 *A Driver Set in an Identity Vault*



You can add other driver sets by dragging the Driver Set object from the palette to the Modeler.

From the **General** page, you can specify or change driver set values.

Table 3-5 *Driver Set Settings*

Field	Description
Name	The name of the Driver Set object. For example, `cn=driverset1,o=system`.
Create a new partition on this driver set	We recommend that you select this option. For details, see the OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Linux and OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Windows.
Deploy context	The Identity Vault assigns the default DN container value to all driver sets in LDAP format. If you specify a DN container here on the Driver Set object, that setting takes precedence over the Identity Vault setting. For example, `o=system`. You can manually enter this value or browse for it.

Driver Set Configuration

You can link in Global Configuration objects to the driver set GCVs. This allows you to reuse Global Configuration objects instead of creating multiple GCVs for the driver set.

To add a Global Configuration object:

1. Click **Add**, then browse to and select the Global Configuration object.
2. Click **Apply** to save the change.

You can change the order that the Global Configuration objects are listed by selecting the object, then clicking **Up** or **Down**.

Driver Set Global Configuration Values

Global configuration values (GCVs) are settings that are similar to driver parameters. Global configuration values can be specified for a driver set as well as an individual driver. If a driver does not have a GCV, the driver inherits the value for that GCV from the driver set.

GCVs allow you to specify settings for OpenText Identity Manager features such as password synchronization and driver heartbeat, as well as settings that are specific to the function of an individual driver configuration. Some GCVs are provided with the drivers, but you can also add your own. You can refer to these values in a policy to help you customize your driver configuration.

To view or change the driver set's GCV settings, double-click the driver set. From the Global Configuration Values page, you can add, edit, or remove values, or edit the XML file for the driver set.

Java Environment Parameters

The Java Environment Parameters enable you to configure the Java virtual machine (JVM) on the OpenText Identity Manager server associated with the driver set.

Table 3-6 *Java Environment Parameters Settings*

Field	Description
Classpath Additions	Specifies additional paths for the JVM to search for package (`.jar`) and class (`.class`) files. Using this parameter is the same as using the `java -classpath` command. When you enter multiple class paths, separate them with a semicolon (;) for a Windows JVM and a colon (:) for UNIX/Linux JVMs.
JVM Options	Specifies additional options to use with the JVM. Refer to your JVM documentation for valid options.
Initial Heap Size	Specifies the initial (minimum) heap size available to the JVM. Increasing the initial heap size can improve startup time and performance. Enter a numeric value followed by g, m, or k (case insensitive). If no letter size is specified, the size defaults to bytes. Using this parameter is the same as using the `java -Xms` command. Refer to your JVM documentation for information about the default initial heap size for the JVM.
Maximum Heap Size	Specifies the maximum heap size available to the JVM. Enter a numeric value followed by g, m, or k (case insensitive). If no letter size is specified, the size defaults to bytes. Using this parameter is the same as using the `java -Xmx` command. Refer to your JVM documentation for information about the default maximum heap size for the JVM.

Driver Set Log Levels

The **Driver Set Log Level** options enable you to view high-level information. For lower-level information, use the **Trace** option.

By default, logging is turned off. To track errors, messages, or events, change the default.

1. Double-click the driver set.
2. Select **Driver Set Log Level.**
3. Select a logging option.

   The log option that you select determines which messages are available in the log.
4. To configure audit instrumentation, select **Log specific events**, click the event selector button, select events, then click **OK**.

   The **Update only the last log time** option updates the time stamp to indicate the last activity of the driver.
5. Specify the number of entries in the log.

   The default is 50 entries (lines) in the log. If you want a longer history, increase the number.
6. Save changes by clicking **OK**.

The driver set log contains messages from the engine when it tries to start or stop drivers. To view the log, use iManager. Select the **Status Log** icon above the Identity Vault in the **OpenText Identity Manager Overview**.

Driver Set Named Passwords

The Named Passwords property page allows you to manage (add, edit, delete) named passwords for the selected driver set. When named passwords are defined in the driver set, the passwords are available to all drivers in the driver set.

NOTE: If you create a named password of the same name in both the driver set and a driver in the driver set, the named password settings in the driver take precedence.

You can define named passwords on both drivers and driver sets. For more information about named passwords, see “Driver Named Passwords” on page 102.

Driver Set Packages

The **Packages** option allows you to manage any packages at the driver set level. A package at the driver set level is applied to all of the drivers that reside in the selected driver set.

The following table lists the options available to manage packages. For more information about packages, see Chapter 6, “Understanding Packages,” on page 171.

Table 3-7 Managing Packages Options

Options	Descriptions
Add package	Adds a package to the driver set. You must add a package before you can install a package. Click the Add package icon, then select the package to install and click OK.
Create package	The Create package option is only available if the Enable Package Developer Mode is selected in the Identity Vault Configuration page. Only developers create packages for redistribution.
Package	Lists the name and the current state of the package.
Version	Lists the version of the package.
Upgrades	Indicates that there is a newer version of a package imported into the package catalog, but it has not been installed. The package needs to be upgraded.
Operation	Lists the operations that can be performed on a package. • **Install:** The Install option is only available after a package is added to the driver set. Select Install, then click Apply to install the package. • **Uninstall:** The Uninstall option is only available after a package is installed to the driver set. Select Uninstall, then click Apply to uninstall the package. • **Upgrade:** The Upgrade option is only available if there is a newer version of the package available for installation. Select Upgrade, then click OK to upgrade the package. • **Downgrade:** The Downgrade option is only available if you have upgraded a package and the older package is installed in the package catalog. Select Downgrade, then click OK to downgrade the package. • **Revert Customizations:** The Revert Customizations option is only available if you have made changes to the policies that are installed with a package. Select Revert Customization, then click Apply to remove the customization.

Driver Set Server List

After adding one or more servers to the Identity Vault, you can view or change the driver set's server association.

Select a server in the Available Servers list, then use the arrows to move the server to the Selected Server list. If a server is not in the Available Servers list, you must first add it by editing the Identity Vault properties. See "Configuring Identity Vaults" on page 70.

Driver Set Trace

Although a driver set has nothing to trace, you can add a trace level to a driver set. The **Trace** setting specifies a trace level used with all drivers associated with the driver set.

With the trace set, DS Trace displays OpenText Identity Manager and DirXML events as the engine processes the events. The trace level affects each driver in the driver set. Use the trace level for troubleshooting issues with the drivers when they are deployed. DS Trace displays the output of the specified trace level.

IMPORTANT: You should use the trace level only for testing or for troubleshooting driver issues. Setting a driver trace level on a production driver can cause OpenText Identity Manager server to process events slowly.

To set a driver set's trace characteristics:

1. In the Outline view or Modeler, right-click the driver set, then select **Properties**.
2. In the driver properties, select **Trace** in the left navigation area.
3. On the Trace page, specify the trace settings for the driver set, then click **OK**.

Table 3-8 *Driver Set Trace Settings*

Field	Description
Trace level	The IDM engine supports the following trace levels: ◆ Trace level 0: Displays fatal messages, errors, warnings and successes. ◆ Trace levels 1: Displays informational messages in addition to the information from Trace level 0. ◆ Trace level 2: Displays contents of XML documents in addition to the information from Trace level 1. ◆ Trace level 3: Displays policy information in addition to the information from Trace level 2.
XSL Trace Level	DS Trace displays XSL events. Set this trace level only when troubleshooting XSL style sheets. If you do not want to see XSL information, set the level to 0.
Java Debug Port	Allows developers to attach a Java debugger.
Trace File	When a value is set in this field, all Java information for the driver is written to file. The value for this field is the path for that file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace File Encoding	The trace file uses the system's default encoding. You can specify another encoding if desired.
Trace File Size Limit	Sets a limit for the Java trace file. Select **Unlimited** to allow the file to grow to fill the disk.

The following methods help you capture and save OpenText Identity Manager trace information.

- “Windows” on page 82
- “UNIX” on page 82
- “iMonitor” on page 82

Windows

Open the Control Panel, select **NDS Services**, then click **DS Trace.DLM** > **Start**. A window named NDS Server Trace Utility opens.

To set the filters to capture the DirXML trace information:

1. Click **Edit** > **Options** > **Clear All**.
2. Click the boxes next to **DirXML** and **DirXML Drivers**, then click **OK**.

To save the information to a file:

1. Click **File** > **New**.

   A dialog box prompts for a filename.
2. Enter a filename with the extension of `.log`.
3. To stop capturing information, click **File** > **Close**.

   The file is saved.

UNIX

Use the `ndstrace` command at the console to display the OpenText Identity Manager events. The `exit` command quits the trace utility.

Table 3-9 *ndstrace Commands*

Command	Description
`Set ndstrace=nodebug`	Turns off all trace flags.
`Set ndstrace on`	Displays trace messages to the console.
`Set ndstrace file on`	Captures trace message to the `ndstrace.log` file in the `/var/nds` directory.
`Set ndstrace file off`	Stops capturing trace messages to the file.
`Set ndstrace=+dxml`	Displays the OpenText Identity Manager events
`Set ndstrace=+dvrs`	Displays the OpenText Identity Manager driver events.

iMonitor

Use iMonitor to get DS Trace information from a Web browser.

Table 3-10 *Platforms and Commands for Web Browsers*

Platform	Command
Windows	`ndsimon.dlm`
Linux/Solaris/AIX/HP-UX	`ndsimonitor`

1. Access iMonitor from http://server_ip:8008/nds (the default port).
2. Click **Trace Configuration**.
3. Click **Clear All**.
4. Click **DirXML** and **DirXML Drivers**.
5. Click **Trace On**, then click **Trace History**.
6. Click the **Current** document icon to view the live trace.

Configuring Libraries

The Library object is a repository of commonly used policies that can be referenced from multiple locations. You can place a policy in the library that every driver in the driver set can reference. You can find the Library object in the Outline view.

The following table lists settings for libraries:

Table 3-11 *Library Settings*

Field	Description
Name	The name of the library. You can modify the name to be more descriptive, especially if you have more than one library in a tree. For example, you might have one library at the Identity Vault level containing policies that are generic to most drivers, and another library at the Driver Set level containing policies that are specific to that driver set.
Deploy Context	The Identity Vault assigns the default DN container value to a library created or deployed at the Identity Vault level. If you specify a DN container here on the Library object, that container setting takes precedence over the Identity Vault setting. You can manually enter this value or browse to and select the context. Libraries created under the driver set do not have the **Deploy Context** option.
Description	This field allows you to type a description of the selected library.

For more information about what you can add to a library, see “Library Objects” in “Library Objects”*OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

Configuring Drivers

A driver provides the connection between an application and the Identity Vault. The driver is the connector that enables data synchronization and sharing between systems. To view or change settings, double-click a driver or driver line in the Modeler.

- "Driver General Settings" on page 84
- "Driver Configuration" on page 85
- "Engine Control Values" on page 87
- "Driver Global Configuration Values" on page 91
- "Driver Health Configuration" on page 93
- "Driver Log Level" on page 101
- "Driver Manifest" on page 102
- "Driver Named Passwords" on page 102
- "Driver Packages" on page 102
- "Reciprocal Attributes" on page 104
- "Driver Trace Levels" on page 107
- "Driver Icon" on page 110

Driver General Settings

The following table contains a description of the general settings for drivers.

Table 3-12 *General Settings*

Field	Description
Name	Displays the driver name, which you can change.
Notes	Enables you to type notes about your driver implementation.
Server/Driver Version	Displays the server name to which driver is associated. The driver version only shows if the driver is running. Driver versions vary for each driver.
(Deprecated) **Basic configuration file**	The field is populated only if you configured your driver by using a driver configuration file instead of packages. Displays the configuration filename that this driver uses. Contains the filename of the configuration file that was used during import. To view the path to this file, click the information icon next to the filename. You might want to view the file to find out version information. If you haven't yet run the import wizard, this field is set to **None**.

Field	Description
Supported DN format	Displays the format (for example, LDAP) that is supported for each driver. This DN information is important for policy building and simulation. For additional details, click the information icon next to the format field.

Driver Configuration

The driver configuration page is dynamic. Labels and descriptions are dynamically read from the driver configuration information.This information is unique for each driver.

The two required options for every driver are **Driver Configuration** and **GCVs**. With the **Driver Configuration** option selected, fill in the required values and parameters that are necessary to have the driver run in your network environment. However, because each driver contains different values and parameters, you need to consult the driver manual for specific values. Go to the OpenText Identity Manager driver documentation website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/), then select the manual for the driver you are configuring.

- "Driver Module" on page 85
- "Startup Option" on page 86
- "Driver Parameters" on page 86
- "ECMAScript" on page 86
- "Global Configuration" on page 87

Driver Module

Table 3-13 *Driver Module Settings*

Field	Description
Java: Name of the Java class	Specify the name of the Java class that will be instantiated for the shim component of the driver. This class can be located in the `classes` directory as a class file, or in the `lib` directory as a `.jar` file.
Native: Name of the DLL	Specify the name of the `.dll` file that will be instantiated for the application shim component of the driver.
Connect to Remote Loader	Select this option if you want to connect the driver to the OpenText Identity Manager engine that uses the Remote Loader.
Driver object password: Set Password	Set a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page or the remote driver cannot run. The Remote Loader uses this password to authenticate itself to the remote driver.

Field	Description
Remote Loader client configuration for documentation: Include in documentation	Enables you to document your Remote Loader configuration for the driver. From the drop-down list, select a name that you specified on the driver's documentation property page. To use this option, see "Engine Control Values" on page 87.

Startup Option

Table 3-14 *Startup Settings*

Setting	Description
Auto start	The driver starts automatically when the OpenText Identity Manager engine loads.
Manual	You must start the driver manually from the driver state location.
Disabled	Disables the driver.
Do not automatically synchronize the driver	If you don't select this option, a driver that has been deployed but disabled resynchronizes on startup. If you select this option, a driver that has been deployed but disabled does not resynchronize.

Driver Parameters

From this tab, you can enter common driver options, Subscriber and Publisher channel options, as well as edit XML. Because the Driver Parameters options are different for each driver, refer to the OpenText Identity Manager driver documentation website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/) for configuration information about the driver you have selected.

ECMAScript

Displays an ordered list of ECMAScript resource files that are loaded when the driver starts. The ECMAScript files contain extension functions that can be used in policies.

To add an ECMAScript from another driver:

1. Click **Add**, then browse to and select the ECMAScript object from another driver.
2. Click **OK**.
3. Click **Apply** to save the change.

For more information, see "Using ECMAScript in Policies" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

Global Configuration

You can link in Global Configuration objects to extend GCV definitions for the driver that OpenText Identity Manager loads when the driver starts. This allows you to reuse Global Configuration objects instead of creating multiple GCVs for the driver.

To add a Global Configuration object:

1. Click **Add**, then browse to and select the Global Configuration object.
2. Click **Apply** to save the change.

You can change the order that the Global Configuration objects are listed by selecting the object, then clicking **Up** or **Down**.

Engine Control Values

The engine control values enable you to change certain default behaviors of the OpenText Identity Manager engine. You can access the values only if a server is associated with the Driver Set object. The values are populated based on the OpenText Identity Manager version of the servers that are associated with the driver set (servers can be associated through the **Engine Controls for Server** entry).

Changing a version of an OpenText Identity Manager server affects the engine controls for all drivers in a driver set that is associated with the server. When the OpenText Identity Manager version is changed, the engine controls for all associated drivers are updated to match the specified version. During the update process, all current settings for existing engine controls are merged into the new engine controls. If the engine controls are not valid for the version of the selected server, they are removed as options.

1. In the Modeler, right-click the driver line.
2. Select **Properties > Engine Control Values**.
3. Click the tooltip icon to the right of the **Engine Controls for Server** field. If a server is associated with the Identity Vault, and if you are authenticated, the engine control values display in the large pane.

Table 3-15 *Engine Control Values*

Field	Description
Subscriber channel retry interval in seconds	Controls how frequently the OpenText Identity Manager engine retries the processing of a cached transaction after the application shim's Subscriber object returns a retry status.
Qualified form for DN-syntax attribute values	Controls whether values for DN-syntax attribute values are presented in unqualified slash form or qualified slash form. When the control is set to True, the attribute values are presented in qualified form.
Qualified form from rename events	Controls whether to present the new-name portion of rename events coming from the Identity Vault to the Subscriber channel with type qualifiers. For example, CN=. When the control is set to True, the names are presented in qualified form.

Field	Description
Maximum eDirectory replication wait time in seconds	Controls the maximum time that the engine waits for a particular change to replicate between the local replica and a remote replica. This only affects operations where the engine is required to contact a remote OpenText eDirectory server in the same tree to perform an operation and might need to wait until some change has replicated to or from the remote server before the operation can be completed. For example, object moves when the OpenText Identity Manager server does not hold the master replica of the moved object or the file system rights operations for users created from a template.
Use non-compliant backwards-compatible mode for XSLT	Sets the XSLT processor used by the engine to a backward-compatible mode. This allows the XSLT processor to use non- XPath 1.0 and XSLT 1.0 standards-compliant behaviors. This is required for backward compatibility with existing OpenText Identity Manager style sheets that depend on the non-standard behaviors. For example, the behavior of the XPath "!=" operator when one operand is a node set and the other operand is other than a node set is incorrect in DirXML releases up to and including Identity Manager 2.0. This behavior has been corrected; however, the corrected behavior is disabled by default through this control to allow backwards compatibility with existing DirXML style sheets.
Maximum application objects to migrate at once	Limits the number of application objects that the engine requests from an application during a single query that is performed as part of a Migrate Objects from Application operation. If java.lang.OutOfMemoryError errors are encountered during a Migrate from Application operation, this number should be set lower than the default. The default is 50. **NOTE:** This control does not limit the number of application objects that can be migrated; it merely limits the batch size.
Set creatorsName on objects created in Identity Vault	Determines whether engine sets the creatorsName attribute to the DN of a driver on all objects created in the Identity Vault by the driver. Setting this attribute allows for easy identification of objects created by the driver and also carries a performance penalty. In absence of a value, the attribute defaults to the DN of the NCP Server object that is hosting the driver.
Write pending associations	Determines whether the engine writes a pending association on an object during Subscriber channel processing. Writing a pending association confers little or no benefit but incurs a performance penalty. Nevertheless, the option exists to turn it on for backwards compatibility.

Field	Description
Use password event values	Determines the source of the value reported for the nspmDistributionPassword attribute for Subscriber channel add and modify events. When this control is set to False, the current value of the nspmDistributionPassword is obtained and reported as the value of the attribute event. This means that only the current password value is available. This is the default behavior. When the control is set to True, the value recorded with the OpenText eDirectory event is decrypted and is reported as the value of the attribute event. Both the old password value (if it exists) and the replacement password value at the time of the event are available. This is useful for synchronizing passwords to certain applications that require the old password to enable setting a new password.
Enable password synchronization status reporting	Determines whether the engine reports the status of Subscriber channel password change events. Reporting the status of Subscriber channel password change events allows applications such as the OpenText Identity Manager User Application to monitor the synchronization progress of a password change that should be synchronized to the managed application.
Combine values from template object with those from add operation	Determines how the engine uses values from a template object when the template is used to create objects in the Identity Vault. The default value is True. The multi-valued attribute values from the template are used in addition to the values for the same attribute that are specified in the add operation. When the control is set to False, the values from the template are ignored if the values for the same attribute are specified in the add operation.
Allow event loopback from publisher to subscriber channel	Determines whether the engine allows an event to loopback from the Publisher channel of a driver to the Subscriber channel of the same driver. The default value is False. The events are not looped back into the Subscriber channel of the driver. When the control is set to True, the events loopback into the Subscriber channel of the driver.
Revert to calculated membership value behavior	Prior to Identity Manager 3.6, the engine retrieved calculated values for Member and Group Membership attributes. The default value is false. The Identity Manager 4.5 engine retrieves static values. This behavior is useful for synchronizing Nested Groups. When the control is set to True, the engine reverts to the pre-3.6 behavior. You can read calculated values for Member and Group Membership attributes with post-3.6 default behavior by using "[pseudo].Member" and "[pseudo].Group Membership" special attribute names.

Field	Description
Maximum time to wait for driver shutdown in seconds	Determines the maximum time in seconds for which the engine waits for the driver's Publisher channel to shutdown. If the driver does not shutdown within the provided time value, the driver is terminated by the engine.The default value is 60 seconds.
Regular Expression escape meta-characters	Determines the meta-characters that will be escaped while evaluating regular expressions. If a meta-char is not present in the value, the character is not escaped during local variable expansion containing a regular expression. To escape all the regular expression meta-characters, "\,$,^,.,?,*,+,[,],(,),|" should be added as the value of the control. The default value is $. If you do not want a meta-character to escape, remove the character from the value. The control value should be a valid comma(,) separated list. Otherwise, you might get errors during policy evaluation.
Retry of Out of Band event	Determines whether the engine retries an out of band event when the status is RETRY. This control is false by default. The OpenText Identity Manager engine will not retry the out of band event on a RETRY status. If this control is set to true, the engine retries the out of band event on a RETRY status.
Use Rhino ECMAScript engine	Determines whether the engine uses the Rhino ECMAScript engine. The engine uses Nashorn as the default ECMAScript engine. This control is true by default.
Enable Subscriber Service Channel	Determines whether the engine processes the out of band queries, such as code map refresh, data collection, and queries triggered from dxcmd, on the Subscriber Service channel of the JDBC Fan-Out driver. By default, this control is set to true. The channel separately processes these queries without interrupting the normal processing of events.
Ignore Entitlement Changes of other drivers	This control determines whether the engine ignores or processes entitlement changes of other drivers. The default value is true. This means that the driver automatically ignores the entitlement changes of other drivers. If this control is set to false, the entitlement changes of other drivers are cached and processed by this driver.
Allow Entitlement event loopback from cprs to subscriber channel	This control determines whether the engine allows an entitlement event that is generated by a CPRS assignment to loopback to the Subscriber channel of the driver. The default value is false. This means that the event is not looped back to the Subscriber channel. If this control is set to true, the event flows to the Subscriber channel of the driver.

Field	Description
Retrieve Application Schema	This control determines whether the driver has to query for the application schema or not. By default, this control is set to true. If you do not want the driver to query for the application schema, set the value to false.

Driver Global Configuration Values

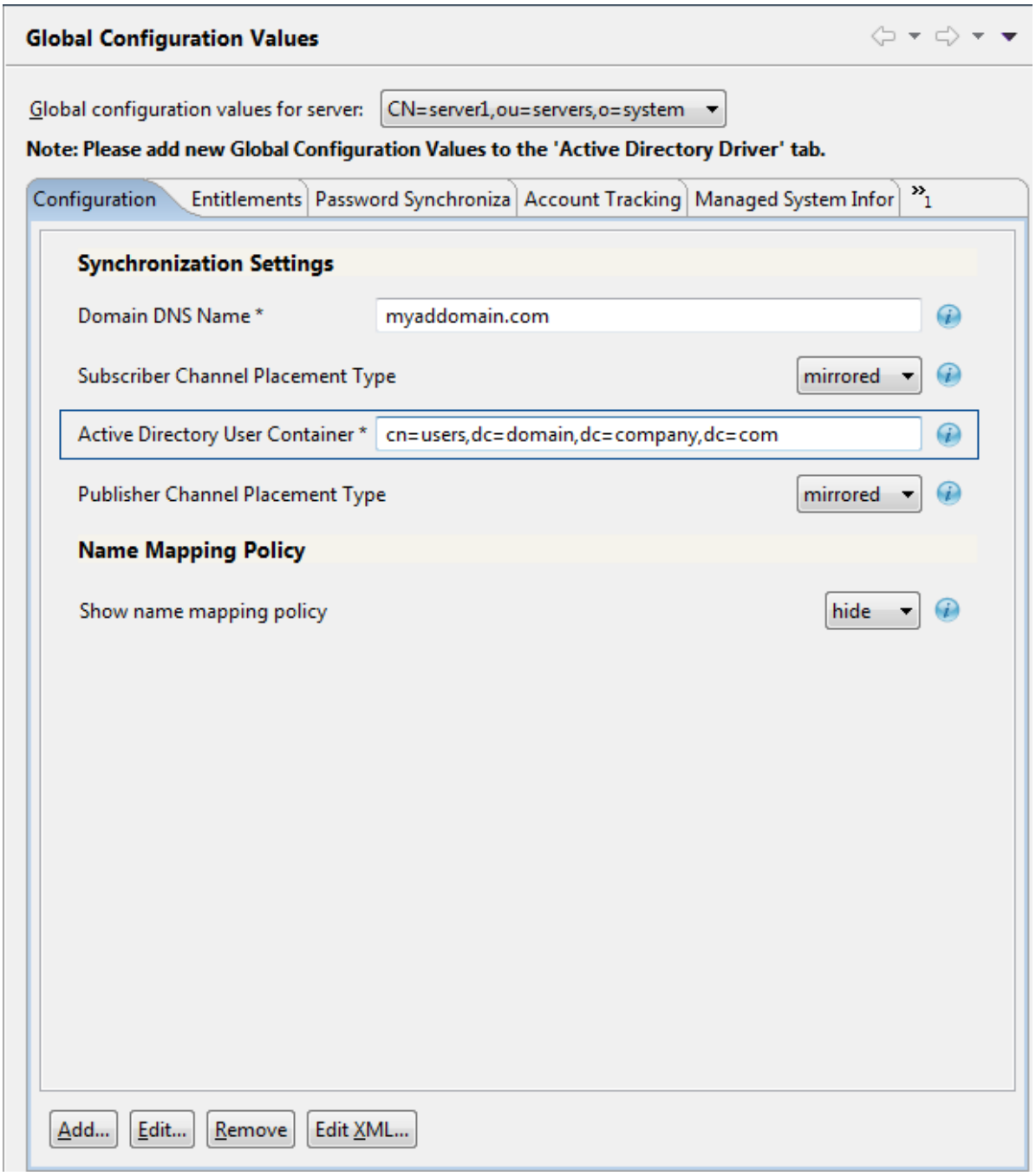
Global configuration values (GCVs) are settings that are similar to driver parameters. GCVs can be specified for an individual driver as well as a driver set. If a driver does not have a GCV, the driver inherits the value for that GCV from the driver set.

GCVs allow you to specify settings for OpenText Identity Manager features such as password synchronization and driver heartbeat, as well as settings that are specific to the function of an individual driver configuration. Some GCVs are provided with the drivers, but you can also add your own. You can refer to these values in a policy to help you customize your driver configuration.

To edit the driver set's GCV settings, double-click the Driver Set object in the Modeler view. From the Global Configuration Values page, you can add, edit, remove, or edit the XML for GCVs.

To view or change the driver's GCV settings, double-click the driver. From the Global Configuration Values page, you can add, edit, or remove values, or edit the XML file for the driver. To select a value, click the value or the control field to the right of the value's name. Use the **Add**, **Edit**, **Remove**, and **Edit XML** buttons at the bottom of the page.

Figure 3-3 *The Global Configuration Values Page*



You can add, edit, and remove GCVs on the Global Configuration Values page, except for those values found under the Password Management heading. Password values are accessed through the Password Synchronization page; click the **Launch Password Sync Dialog** icon to the right of the **Information** icon for the control field.

The two required options for configuring a driver are **Driver Configuration** and **GCVs**. However, because each driver contains different values and parameters, you need to consult the driver manual for specific values. Go to the OpenText Identity Manager driver documentation website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/), then select the manual for the driver you are configuring.

Driver Health Configuration

The Driver Health Configuration allows you to monitor a driver's state of health (green, yellow, or red), and to specify the actions to perform in response to each of these health states.

To do so, you define the conditions (criteria) that determine each of the health states, and the associated actions to perform whenever the driver's health state changes. For example, if the driver's health changes from a green state to a yellow state (based on the conditions you establish), you can perform such actions as restarting the driver, shutting down the driver, and sending an e-mail to the person designated to resolve issues with the driver.

You can also define custom driver states that are independent of the standard green, yellow and red. Whenever the driver meets the conditions for the custom state, Designer performs the associated actions.

To use the Driver Health Configuration to monitor a driver's health state, you must complete the following tasks:

- "Creating a Driver Health Configuration" on page 93
- "Modifying the Health State Conditions" on page 94
- "Creating a Driver Health Job" on page 97

Additionally, you can perform the following tasks to further configure the Driver Health Check environment:

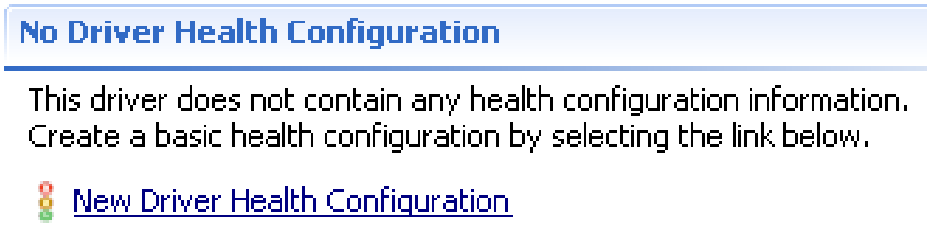
- "Modifying the Health State Actions" on page 98
- "Creating a Custom State" on page 99
- "Modifying the Driver Health Job Settings" on page 100

NOTE: Monitoring driver health is applicable only to deployed drivers. Designer does not indicate driver health in the Modeler or any other pre-deployment interface. After you set up the health configuration, you use iManager to actually monitor the health of deployed drivers. For more information about driver health monitoring in iManager, see "Monitoring Driver Health" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

Creating a Driver Health Configuration

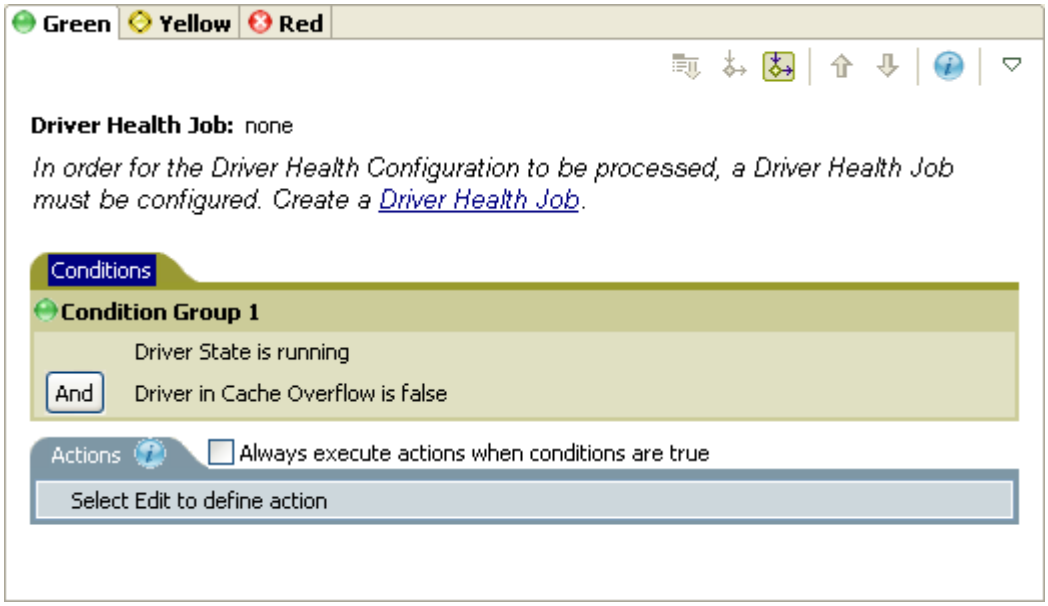
The health configuration of drivers is configured automatically, unless you are running older versions of OpenText Identity Manager. If you are running anything older than Identity Manager 3.6, you must complete the following section to create a driver health configuration. Otherwise, skip this section.

1. In the Modeler or Outline view, right-click the driver, then select Properties.
2. In the left-side navigation, select **Health**.



3 Select **New Driver Health Configuration**.

Designer creates a basic health configuration with sample conditions for the green and yellow states (none for red).



4 Continue with "Modifying the Health State Conditions" on page 94.

Modifying the Health State Conditions

The driver health configuration lets you define the conditions that determine each health state. The green state contains conditions intended to represent a healthy driver, and a red state represents an unhealthy driver that has failed the conditions for both green and yellow states.

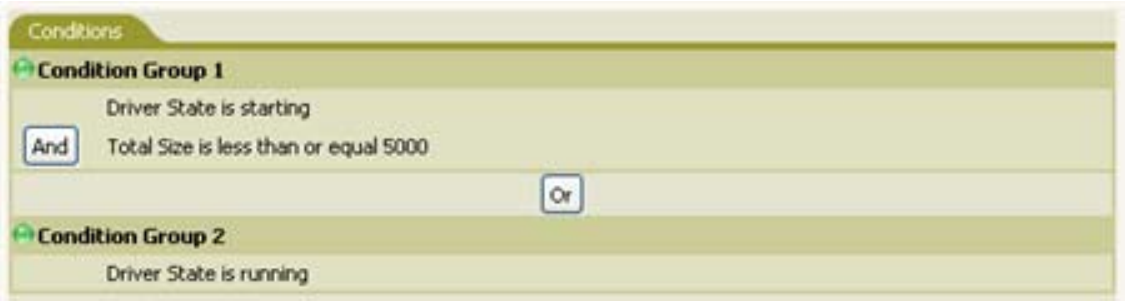
The Driver Health job evaluates the conditions for the green state first. If the driver fails to meet the green conditions, it evaluates the yellow conditions. If the driver fails to meet the yellow conditions, it is automatically assigned a red state.

To modify the conditions for a state:

1 In the Modeler or Outline view, right-click the driver where you want to modify the health check configuration, then select **Properties**.

2 In the left-side navigation, select **Health**.

3 Click the state tab (Green or Yellow) that you want to modify.



The tab displays the current conditions for the health state. Conditions are organized into groups, with logical operators (either AND or OR), to link each condition and condition group.

Table 3-16 describes the conditions that the Driver Health job can evaluate.

Table 3-16 *Driver Health Check Conditions*

Condition	Description
Driver State	Running, stopped, starting, not running, or shutting down. For example, one of the default conditions for the green health state is a Driver State that indicates the driver is running.
Driver in Cache Overflow	The state of the cache used for holding driver transactions. If the driver is in cache overflow, all available cache has been used. For example, the default condition for the green health state is Driver in Cache Overflow is false and the default for the yellow health state is Driver in Cache Overflow is true.
Newest	The age of the newest transaction in the cache.
Oldest	The age of the oldest transaction in the cache.
Total Size	The size of the cache in bytes.
Unprocessed Size	The size of all unprocessed transactions in the cache.
Unprocessed Transactions	The number of unprocessed transactions in the cache. You can specify all transactions types or specific transaction types (such as adds, removes, or renames).

Condition	Description
Transaction History	The number of transactions processed at various points in the Subscriber or Publisher channel over a given period of time. This condition uses multiple elements in the following format: <*transaction type*> <*transaction location and time period* > <*relational operator*> <*transaction number*>. ◆ <*transaction type*>: Specifies the type of transaction being evaluated. For example, adds, removes, renames, and so forth. ◆ <*transaction location and time period*>: Specifies the point in the Subscriber or Publisher channel and the time period being evaluated. For example, you might evaluate the total number of transactions processed as Publisher events over the last 48 hours. The time period cannot exceed the Transaction Data Duration setting, which is configurable in the Driver Health job. For more information, see "Modifying the Driver Health Job Settings" on page 100. ◆ <*relational operator*>: Specifies the relationship between the identified transactions and the *<transaction number>* (equal to, less than, greater than, and so forth.) ◆ <*transaction number*>: Specifies the number of transactions being used in the evaluation. For example: `<number of adds> <as publisher commands> <over the last 10 minutes> <is less than> <1000>`
Available History	The amount of transaction history data that is available for evaluation. This condition helps ensure that a Transactions History condition does not cause the current state to fail because it does not have enough transaction history data collected for the time period being evaluated. For example, assume that you want to use the Transactions History condition to evaluate the number of "Add as Publisher" commands over the last 48 hours. However, you don't want the condition to fail if there is less than 48 hours of data. You could create condition groups similar to the following: `Group1 Available History <is less than> <48 hours> or Group2 Available History <is greater than or equal to> <48 hours> and Transactions History <number of adds> <as publisher commands> <over the last 48 hours> <is less than> <1000>` The state evaluates to true if either condition group is true. The state evaluates to false if both conditions evaluate to false.

4 Modify the condition criteria as desired.

- To add a new group, select the **Conditions** tab, then click **Append Condition Group**  .
- To add a condition, select an existing condition group, then click **Append Condition**  .

- To reorder condition groups or individual conditions, select the condition group or condition, then click **Move Up** or **Move Down**. You can also use these buttons to move a condition from one group to another.
- Cut, copy, and paste a condition group or condition to the clipboard by right-clicking the item, then selecting the appropriate clipboard action.

5 Click **Apply** to save your changes without closing the Properties page, or click **OK** to save the changes and close the Properties page.

6 If you want to change the actions associated with the conditions you set, continue with "Modifying the Health State Actions" on page 98.

Creating a Driver Health Job

The Driver Health job executes periodically to evaluate the health of a driver configured for health checks. The job evaluates the conditions defined for each of the driver's health states, then assigns the driver the appropriate state. The job also executes any actions associated with the assigned state.

If a Driver Health job does not exist, the Driver Health Configuration page displays a **New Driver** link from which you can configure the Driver Health job. If a Driver Health job already exists, the Driver Health Configuration page does not display this prompt.

To create a Driver Health job:

1 In the Modeler or Outline view, right-click the driver, then select **Properties**.

2 In the left-side navigation, select **Health**.

3 Click **Driver Health Job** to open the Job dialog box. Select the appropriate job, then click **OK**.

Follow the prompts to import the configuration file for the Driver Health job. Refer to the following information for details:

- **Where to place the driver:** Place the job in the same driver set as the driver. The correct driver set is selected by default. You can only have one Driver Health job per driver set.
- **Import a configuration:** Import the configuration from the server. In the **Show** field, select Identity Manager 4.7 configurations, then select the Driver Health job in the **Configurations** field.
- **Email server:** Select the e-mail server that you want used for any actions that initiate e-mail. If you have not defined additional e-mail servers, select the Default Notification Collection server.
- **Servers:** If the driver set is associated with only one server, that server is selected and cannot be changed. If the driver set is associated with multiple servers, select the server where you want to run the job.

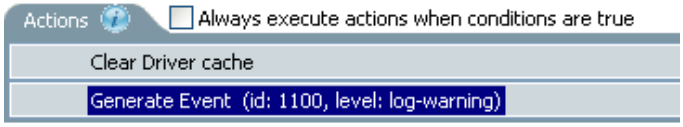
After creating the Driver Health job, you can modify job settings as needed. For example, you can configure how often the job runs, which drivers use the job, and how much data the job maintains to support transaction history. For more information, see "Modifying the Driver Health Job Settings" on page 100.

Modifying the Health State Actions

The Driver Health Configuration lets you define the actions that the Driver Health job performs when the driver health state changes. For example, if the state changes from green to yellow, you can shut down or restart the driver, generate an event, or start a workflow.

The Driver Health job performs a health state's actions only once each time the conditions are met; as long as the driver state remains the same, the actions do not repeat. If the driver state changes because its conditions are no longer met, the Driver Health job performs the state's actions again the next time its conditions are met.

1. In the Modeler or Outline view, right-click the driver where you want to modify the health check configuration, then select **Properties**.
2. In the left-side navigation, select **Health**.
3. Select the state tab (Green or Yellow) that you want to modify.



   The tab displays the current actions for the health state. If no action is assigned, the Driver Health Configuration displays **Define new action here** in the **Actions** tab.
4. Select the **Actions** tab, then click **Append Action** to add an action to the health state.
5. Select an action from the drop-down list.The table below describes the actions that the Driver Health job can perform.

   Some actions require additional information before they will execute.

Action	Description
Clear Driver Cache	Removes all transactions, including unprocessed transactions, from the cache.
Execute ECMAScript	Executes an existing ECMAScript. Specify the DirXML-Resource object that contains the ECMAScript.
Generate Event	Generates an event that can be used by Sentinel and the Identity Reporting Module.
On Error	If an action fails, this action tells Designer what to do with the remaining actions, the current health state, and the Driver Health job.
Restart Driver	Restarts the driver (stop, then start)
Send Email	Sends an e-mail to one or more recipients. The template you want used in the e-mail message body must already exist.
Start Driver	Starts the driver.
Start Workflow	Starts a provisioning workflow. For more information about the Start Workflow action, see "Start Workflow" in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies Guide*.
Stop Driver	Stops the driver.
Write Trace Message	Writes a message to the Driver Health job's log file or the driver set's log file if the trace file is not configured on the Driver Health job.

6 Click **Apply** to save your changes without closing the Properties page., or click **OK** to save the changes and close the Properties page.

Creating a Custom State

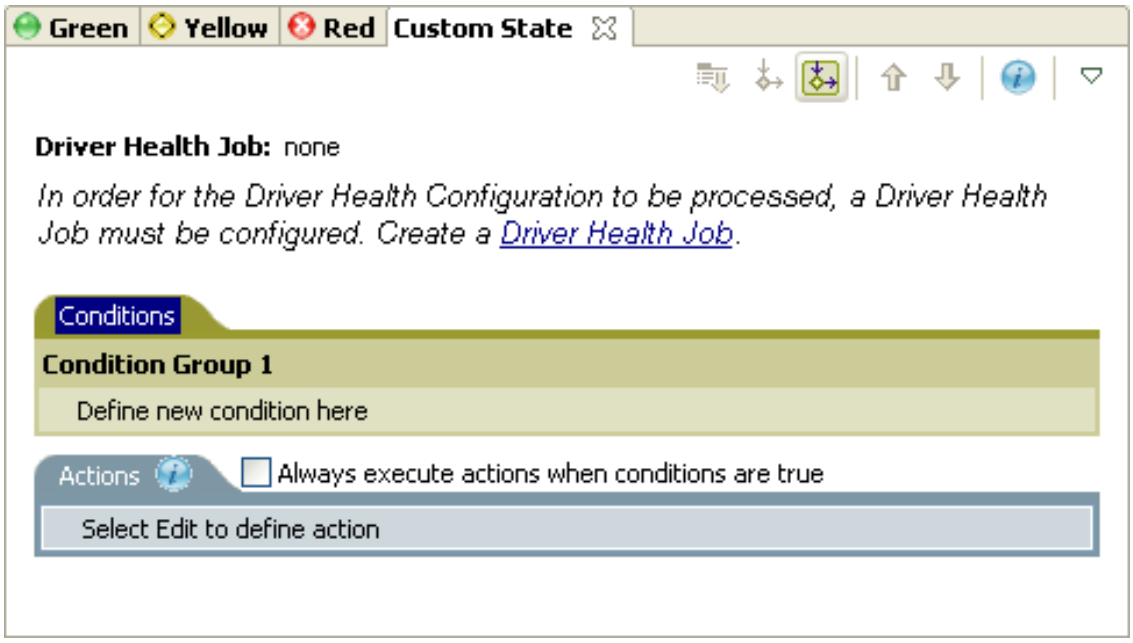
The Driver Health Configuration lets you create one or more custom states to perform actions independent of the driver's current health state (green, yellow, red). If the driver meets the custom state's conditions, the Driver Health job performs its actions.

As with the standard driver health states (green, yellow, red), the Driver Health job performs a custom state's actions only once each time the conditions are met; as long as the driver state remains the same, the actions do not repeat. If the driver state changes because the custom state's conditions are no longer met, the Driver Health job performs the custom state's actions again the next time its conditions are met.

1 In the Modeler or Outline view, right-click the driver where you want to create a custom state, then select **Properties**.

2 In the left-side navigation, select **Health**.

3 Select the drop-down menu , then select **New Custom State**.



4 Define the conditions and actions for the custom state, then click **Apply** to save the changes without closing the Properties page, or click **OK** to save the changes and close the Properties page.

For information about defining state conditions, see "Modifying the Health State Conditions" on page 94. For information about defining state actions, see "Modifying the Health State Actions" on page 98.

Modifying the Driver Health Job Settings

The Driver Health job evaluates the conditions for the health states and assigns the driver the appropriate state. The job also executes any actions associated with the assigned state.

As with all driver jobs, there are several settings that you can modify to optimize the job's performance for your environment, including how often the job runs, which drivers use the job, and how much data the job maintains to support transaction history.

1 In the Modeler or Outline view, open the driver set object where the driver health job is stored.

2 Right-click the appropriate job object, then select **Edit**.

3 Change the desired settings on the following tabs, then click **OK** to save your changes:

Tab	Description
Schedule	The Driver Health job is a continuously running job, meaning that it does not stop unless a health state action shuts it down or you shut it down manually. The job must run continuously to be able to support transaction data collection for use in Transactions History conditions. If the job does stop, it is restarted based on the schedule. The default schedule checks every minute to see if the job is running. If the job is not running, it is started.

Tab	Description
Scope	By default, the job applies to all drivers in the driver set. This means that you only need one Driver Health job per driver set. However, you can create multiple Driver Health jobs for different drivers within the same driver set. For example, you might have some drivers whose health you want updated more frequently than other drivers, in which case you would need at least two Driver Health jobs.
Parameters	You can change any of the following job parameters: ◆ **Login ID:** This defaults to the login ID that was used when creating the driver job. You should only change this if you want the driver to authenticate using different credentials. ◆ **Login password:** This is the password required for the login ID that you supplied in the **Login ID** field. ◆ **Polling interval:** Determines how often the job evaluates the conditions for the health states, assigns the driver the appropriate state, executes any actions associated with the assigned state, and stores the driver's transaction data. The default polling interval is one minute. ◆ **Polling interval units:** Specifies the time unit (minutes, hours, days, weeks) for the number specified in the **Polling interval** setting. ◆ **Duration transaction data is kept:** Specifies how long a driver's transaction data is kept. The default retains a transaction for two weeks before being deleted. Longer transaction durations require more memory. For example, to store transaction data for one driver every minute (**Polling interval**) for two weeks requires approximately 15 MB of memory. ◆ **Duration units:** Specifies the time unit (minutes, hours, days, weeks) for the number specified in the **Duration transaction data is kept** setting.

Driver Log Level

The Driver Log Level options enable you to view high-level information. For lower-level information, use the Trace option. See "Driver Trace Levels" on page 107.

By default, logging inherits the setting from the driver set. To change the default:

1. Right-click the driver and select **Driver > Properties**.
2. Select **Log Level.**
3. Select a logging option.

   The option that you select determines which information is available in the log.
4. To configure the audit instrumentation, select **Log specific events**, click the event selector button, select events, then click **OK**.
5. Specify the number of entries in the log.

   The default is 50 entries (lines) in the log. If you want a longer history, increase the number.
6. Save changes by clicking **OK**.

The driver log contains messages from the driver. The messages are related to operations that the driver performed or tried to perform. To view the log, use iManager. Select the log icon on the Driver object in the OpenText Identity Manager Overview.

Driver Manifest

The driver manifest is like a resume for the driver. The driver manifest states what the driver supports, and includes a few configuration settings. The driver developer should provide the driver manifest. Usually a network administrator does not need to edit the driver manifest.

For more information, see the developer documentation for OpenText Identity Manager drivers.

Driver Named Passwords

The Named Passwords property page allows you to manage (add, edit, delete) named passwords for the selected driver. You can define named passwords on both drivers and driver sets.

Named passwords let you store multiple passwords securely by referring to each password by a key, or name. When you refer to the named password in a driver policy, you use the name only, not the password value. Then, when the driver needs the password value to execute the policy, it requests the password value from the engine. This method lets you avoid revealing the password value in the code for a driver policy.

The following example shows how a named password can be referenced in a driver policy on the Subscriber channel in XSLT:

```
<xsl:value-of
select="query:getNamedPassword($srcQueryProcessor,'mynamedpassword')"
xmlns:query="http://www.novell.com/java/
com.novell.nds.dirxml.driver.XdsQueryProcessor/>
```

You can store and retrieve named passwords for any driver without making changes to the driver shim.

As a security measure, in addition to using named passwords, you should control access to all OpenText Identity Manager objects in OpenText eDirectory. Named passwords are only deployed in a secure connection with OpenText eDirectory.

NOTE: A driver developer can also customize a driver to use named passwords in other ways, such as retrieving named passwords when the driver starts up, instead of requesting them from the engine each time they are needed.

For example, the OpenText Identity Manager Driver for Lotus Notes has been customized to support additional ways of using named passwords, and examples of those methods are included in the sample driver configurations. For more information, see the OpenText Identity Manager driver documentation website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/).

Driver Packages

The **Packages** option allows you to manage any packages at the driver set level. A package at the driver set level is applied to all of the drivers that reside in the selected driver set.

The following table lists the options available to manage packages. For more information about packages, see Chapter 6, “Understanding Packages,” on page 171.

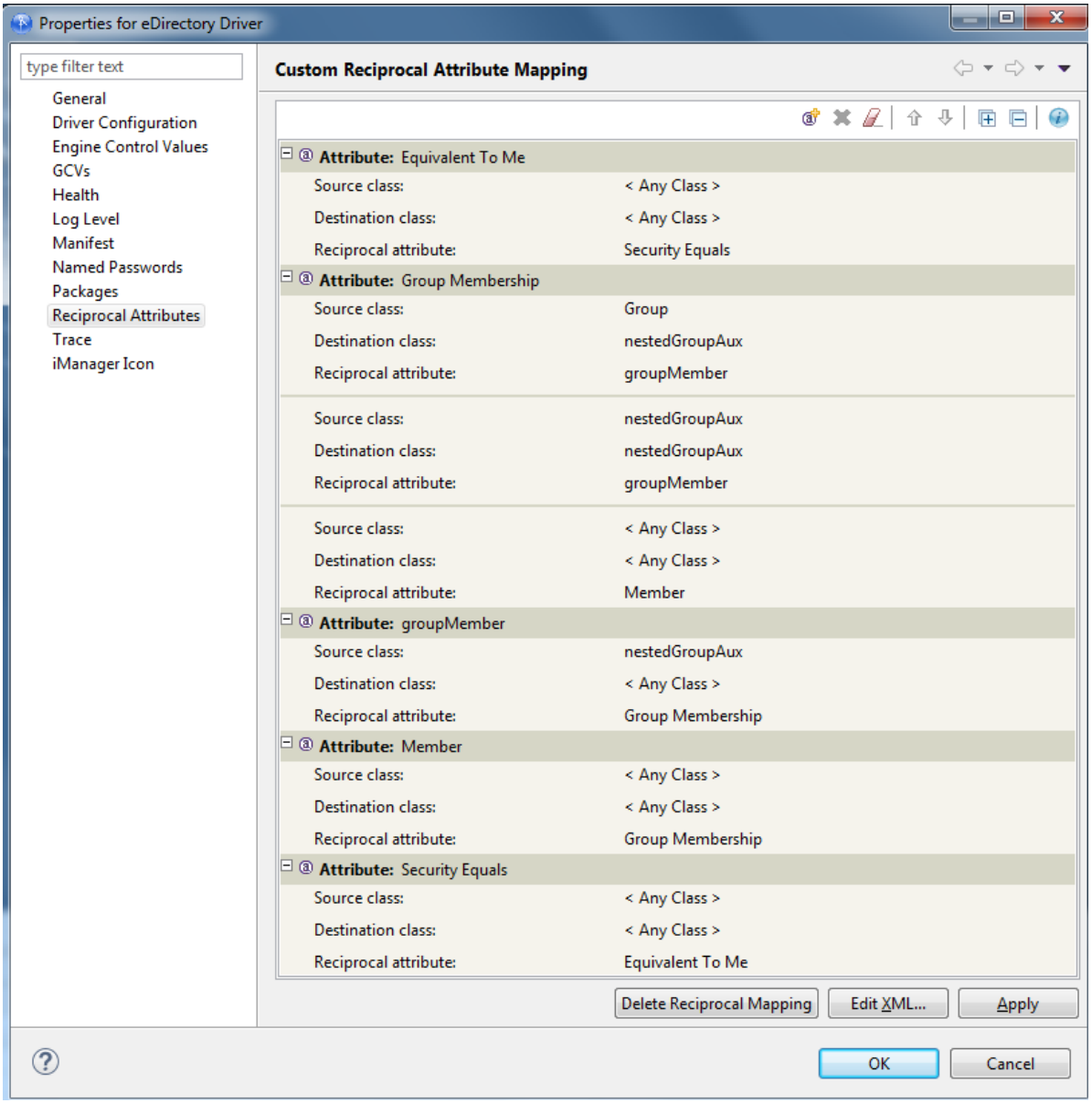
Table 3-17 *Options for Managing Packages*

Options	Descriptions
Add package	Adds a package to the driver. You must add a package before you can install a package. Click the **Add package** icon, then select the package to install and click **OK**.
Create package	The **Create package** option is only available if the **Enable Package Developer Mode** is selected on the Identity Vault Configuration page. Only developers create packages for redistribution.
Package	Lists the name and current state of the package.
Version	Lists the version of the package.
Upgrades	Indicates that there is a newer version of a package imported into the package catalog, but it has not been installed. The package needs to be upgraded.
Operations	Lists the operations that can be performed on a package: ◆ **Install:** This option is only available after a package is added to the driver. Select **Install**, then click **Apply** to install the package. ◆ **Uninstall:** This option is only available after a package is installed to the driver. Select **Uninstall**, then click **Apply** to uninstall the package. ◆ **Upgrade:** This option is only available if there is a newer version of the package available for installation. Select **Upgrade**, then click **OK** to upgrade the package. ◆ **Downgrade:** This option is only available if you have upgraded a package and the older package is installed in the package catalog. Select **Downgrade**, then click **OK** to downgrade the package. ◆ **Revert Customizations:** This option is only available if you have made changes to the policies that are installed with a package. Select **Revert Customization**, then click **Apply** to remove the customization. ◆ **Sync Customizations:** This option is only available if the Enable Package Developer mode is enabled on the Identity Vault and you have made changes to content in a custom package that is installed on this driver. The **Sync Customizations** option synchronizes any changes you have made to the package content to the package. For more information, see Section 7, “Developing Packages,” on page 193.
Run driver in Factory Mode	Allows you to revert any customizations to content installed with packages. For more information, see “Running a Driver in Factory Mode” on page 191.

Reciprocal Attributes

The Reciprocal Attributes property page lets you create and manage backlinks between objects. For example, the Group object includes a Members attribute that contains pointers to all User objects that belong to that group. Similarly, each User object includes a Group Membership attribute that points to the Group objects of which that user is a member. These two-way links between objects are known as *reciprocal mappings*.

Figure 3-4 *Custom Reciprocal Attribute Mapping Property Page for Driver Objects*



You can manage all reciprocal mapping configuration from the toolbar in the property page, which contains the following toolbar icons:

Icon	Description
	Use the **New Attribute** icon to add a new attribute to the reciprocal mapping list.
	Use the **Delete** icon to delete the currently selected reciprocal mapping entry from the list.
	Use the **Clear All Attribute Mappings** icon to delete all reciprocal mappings.
	Use the **Move Up** icon to move the currently selected attribute up in the mapping list. To do so, select the attribute entry you want to move up, then click **Move up**.
	Use the **Move Down** icon to move the currently selected attribute down in the mapping list. To do so, select the attribute entry you want to move down, then click **Move Down**.
	Use the **Expand All** icon to expand all reciprocal attribute mapping entries.
	Use the **Collapse All** icon to expand all reciprocal attribute mapping entries.

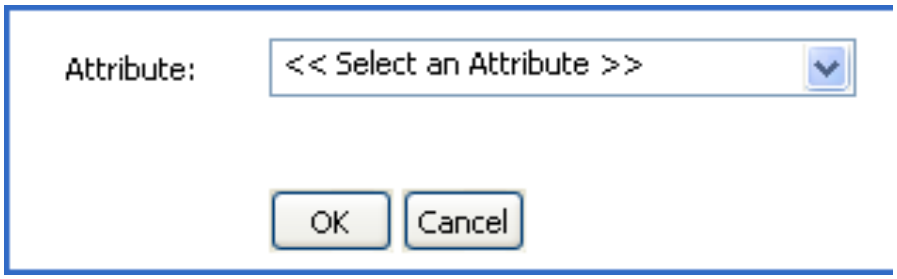
The Custom Reciprocal Mapping page lets you do the following:

- “Adding a Reciprocal Attribute Mapping” on page 105
- “Removing a Reciprocal Attribute Mapping” on page 106
- “Removing an Attribute from the Reciprocal Mapping List” on page 106
- “Editing Reciprocal Attribute XML” on page 107

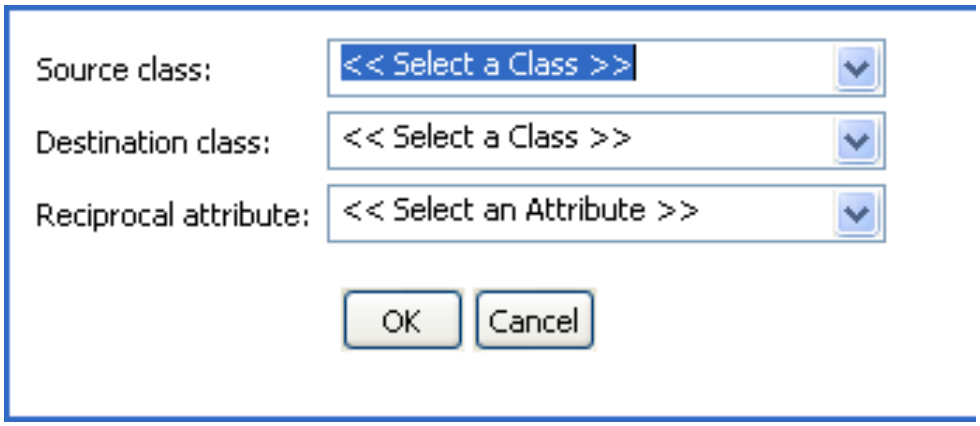
Adding a Reciprocal Attribute Mapping

When you create a reciprocal attribute mapping, you must first add one of the attributes to the reciprocal mapping list:

1. On the Reciprocal Attributes page, click **New Attribute**  .
2. In the new attribute entry, select the desired attribute from the drop-down list, then click **OK**.



3. Specify the details of the reciprocal mapping, then click **OK**.



Source Class	Specifies the class name to which the attribute in the mapping list is associated. For example, if you placed the Group Membership attribute in the reciprocal mapping list, the associated Source Class is User.
Destination Class	Specifies the class name associated with the attribute to which you want to create a reciprocal mapping.
Reciprocal Attribute	Specifies the attribute name to which you want to create a reciprocal mapping.

Removing a Reciprocal Attribute Mapping

To remove a reciprocal mapping between attributes:

1 In the reciprocal mapping list, select the reciprocal mapping you want to remove.

When the mapping is selected, the attribute name in the **Attribute** tab is highlighted.

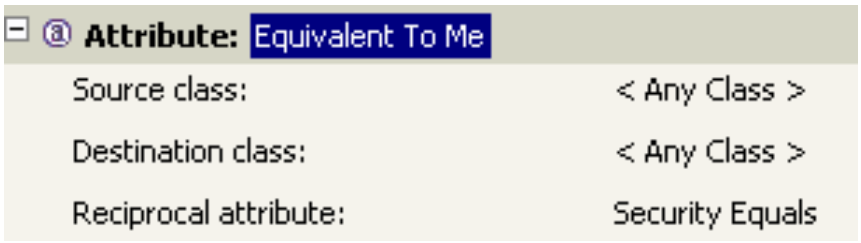


2 Click **Delete** .

Removing an Attribute from the Reciprocal Mapping List

1 Select the attribute you want to remove by selecting it in the reciprocal mapping list.

When selected, the attribute name in the **Attribute** tab is highlighted.



2 Click **Delete** .

To remove all attributes from the reciprocal attribute mapping list, click **Clear All Attribute Mappings** .

Editing Reciprocal Attribute XML

If desired, you can directly edit the XML for a reciprocal attribute. To do so, click **Edit XML** on the Custom Reciprocal Attribute Mapping page. This opens a basic XML editor that lets you modify the XML. When you finish, click **OK** or **Cancel** to close the XML editor.

Driver Trace Levels

You can add a trace to your driver. With the driver trace level set, DS Trace displays driver-related OpenText Identity Manager events, at the level of detail specified by the driver trace level, as the engine processes the events. The driver trace level affects only the driver or driver set where it is set.

IMPORTANT: You should use the trace level only for testing or for troubleshooting driver issues. Setting a driver trace level on a production driver can cause OpenText Identity Manager server to process events slowly.

To set a driver's trace characteristics:

1 In the Outline view or Modeler, right-click the driver, then select **Properties**.

2 In the driver properties, select **Trace** in the left navigation.

3 On the Trace page, specify the driver's trace settings, then click **OK**.

Field	Description
Trace level	The engine supports the following trace levels: ◆ Trace level 0: Displays fatal messages, errors, warnings and successes. ◆ Trace levels 1: Displays informational messages in addition to the information from Trace level 0. ◆ Trace level 2: Displays contents of XML documents in addition to the information from Trace level 1. ◆ Trace level 3: Displays policy information in addition to the information from Trace level 2. Consult the driver documentation for additional trace options that might be available. **NOTE:** You can also set the driver trace level in Designer by right-clicking a driver (in the Outline or Modeler views) and selecting **Live** > **Set Driver Trace Level**. This immediately deploys the trace level to the selected driver. To update the driver trace level in your project as well, select **Update local model**.
Trace level: Use setting from the driver set	If you select this option, all trace levels set at the driver set take precedence over any driver settings. Otherwise, the driver settings are effective.
Trace file	Specify a filename and location where the OpenText Identity Manager information is written for the selected driver. When a value is set in this field, all Java information for the driver is written to file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace file: Use setting from the driver set	If you select this option, all trace levels set at the driver set level take precedence over any driver settings. Otherwise, settings at the driver level are effective.
Trace File Encoding	The trace file uses the system's default encoding. You can specify another encoding if desired.
Trace file size limit	Allows you to set a limit for the Java trace file. Select **Unlimited** to allow the file to grow to fill the disk. **NOTE:** The trace file is created in multiple files. OpenText Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.
Trace file size limit: Use setting from the driver set	If you select this option, all trace levels set at the driver set level take precedence over any driver settings. Otherwise, settings at the driver level are effective.

Field	Description
Trace name	Helps you track trace messages. The name that you specify here appears with the driver trace messages. Use a trace name if the driver name is very long.

The following methods help you capture and save OpenText Identity Manager trace information.

Windows

Open the Control Panel, select **NDS Services**, then click **DS Trace.DLM** > **Start**. A window named NDS Server Trace Utility opens.

To set the filters to capture the OpenText Identity Manager trace information:

1. Click **Edit** > **Options** > **Clear All**.
2. Click the boxes next to **DirXML** and **DirXML Drivers**, then click **OK**.

To save the information to a file:

1. Click **File** > **New**.

   A dialog box prompts for a filename.
2. Enter a filename with the extension of `.log`.
3. To stop capturing information, click **File** > **Close**.

   The file is saved.

UNIX

Use the `ndstrace` command at the console to display the OpenText Identity Manager events. The `exit` command quits the trace utility.

Table 3-18 *ndstrace Commands*

Command	Description
`Set ndstrace=nodebug`	Turns off all trace flags.
`Set ndstrace on`	Displays trace messages to the console.
`Set ndstrace file on`	Captures trace message to the `ndstrace.log` file in the `/var/nds` directory.
`Set ndstrace file off`	Stops capturing trace messages to the file.
`Set ndstrace=+dxml`	Displays the OpenText Identity Manager events
`Set ndstrace=+dvrs`	Displays the OpenText Identity Manager driver events.

iMonitor

Use iMonitor to get DS Trace information from a Web browser.

Table 3-19 Platforms and Commands for Web Browsers

Platform	Command
Windows	`ndsimon.dlm`
Linux/Solaris/AIX/HP-UX	`ndsimonitor`

1. Access iMonitor from http://server_ip:8008/nds (the default port).
2. Click **Trace Configuration**.
3. Click **Clear All**.
4. Click **DirXML** and **DirXML Drivers**.
5. Click **Trace On**, then click **Trace History**.
6. Click the **Current** document icon to view the live trace.

Driver Icon

You can navigate to and select an image for your drivers and applications in the Icon editor. The image formats supported in Designer are PNG, JPG, JPEG, GIF, and BMP.

To browse for driver icons, in the Modeler, right-click an application, click **Properties**, click **iManager Icon**, and then click the **Browse** button to locate the image or icon that you want.

To browse for application icons, in the Modeler, right-click an application, click **Properties**, click **General**, and then click the **Browse** button to locate the image or icon that you want.

Configuring Policies

- "Editing a Policy Name" on page 110
- "Viewing References" on page 110

Editing a Policy Name

1. In the Outline view, right-click a policy or rule.
2. Select **Properties**.

   The **General** setting displays by default.
3. Edit the name in the **Policy Name** field, then click **OK**.

Viewing References

The References page lists policy sets and policies that reference the policy listed in the General page. To view the references to this policy:

1. In the Outline view, right-click a policy or rule.
2. Select **Properties** > **References**.

Linkage is how the policies reference each other. In Identity Manager versions earlier than 3.5, linkage determined the order that policies were executed. To change the linkage, use the Policy Builder.

Configuring Resource Objects

Resource objects store arbitrary data in any format that drivers use. There are different types of Resource objects. For more information, see "Storing Information in Resource Objects" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

The configuration options for Resource objects are:

- **Policy Name:** Stores the name of the resource object. You can change the name.
- **Supported Mime Types:** Allows you to change the type of Resource object. For example, you can change a text Resource object to an XML Resource object.

Configuring Categories

Packages are organized by categories so it is easier to find the packages you need. When you configure the category, you can change the name or add a description.

Configuring Groups

Packages are organized by categories and then groups. This makes finding packages much easier. When you configure the group, you can change the name or add a description.

Configuring Packages

Packages contain OpenText Identity Manager content used to create drivers. You can make configuration changes to packages by right-clicking a package and selecting **Properties**. For more information about packages, see Chapter 6, "Understanding Packages," on page 171.

- "Package General Settings" on page 112
- "Package Configuration Wizard" on page 112
- "Package Constraints" on page 113
- "Package Dependencies" on page 114
- "Package Initial Settings" on page 114
- "Package Languages" on page 114
- "Package License" on page 114
- "Package Linkage" on page 114
- "Package Readme" on page 115
- "Package Targets" on page 115
- "Package Vendor" on page 115

Package General Settings

This property page lists the general settings for the package. These options can be changed only when a package is being developed. After a package is released or imported, these items cannot change.

Table 3-20 *Package General Settings*

Setting	Description
Name	Displays the package name.
Short Name	Displays the unique short name for the package. This name is unique for the package in the Identity Vault.
Version	Displays the package version.
Description	Displays a description for the package.
Type	Lists what type of package it is. It lists whether it is a base package, and if it can be installed on an Identity Vault, driver set, or driver.
Protected	If this option is selected, the **Copy package** option is disabled on imported packages. This allows a developer to protect the content of a package and not allow someone else to create a new package with this content.
Category	Lists the category the package is stored in.
Group	Lists the group the package is stored in.
Meta data	Lists specific information about a package. It lists: ◆ When the package was created. ◆ When the package was built. ◆ If the package is released or not. ◆ If the package has been imported. ◆ Lists where the package is hosted. ◆ Lists the name of the user who built the package.

Package Configuration Wizard

This property page is displayed only on driver base packages. The settings customize what is displayed when users use the Driver Configuration Wizard to install a driver base package.

The Configuration Wizard is an XML editor. Copy the contents of from an existing driver base package that contains the functionality you want to have in this driver base package to this page.

The following is taken from the Active Directory driver base package as an example:

```
<?xml version="1.0" encoding="UTF-8"?><features>
  <mandatory/>
  <optional>
    <group display-name="Default Configuration" expanded="false">
      <package id="5DRKWAWH_201009040020200702" name="Defautl
Configuration" selected="true"/>
    </group>
    <group display-name="Entitlements and Exchange Mailbox Support"
expanded="false">
      <package id="PJP89Z9R_201003031352370466" name="Active Directory
Entitlements and Exchange Mailbox Support" selected="true"/>
      <package id="DETECXTK_201004161538110582" name="Audit Entitlements
Common" selected="true"/>
      <package id="YMO9C1Y3_201006291302430386" name="Active Directory
Audit Entitlements" selected="true"/>
    </group>
    <group display-name="Password Synchronization" expanded="false">
      <package id="XTEF1YO3_201006231733410161" name="Password
Synchronization Common" selected="true"/>
      <package id="4EHOWL6T_201006291417220804" name="Active Directory
Password Synchronization" selected="true"/>
    </group>
    <group display-name="Data Collection" expanded="false">
      <package id="IJLG31AY_201006141353520247" name="Managed System
Information for AD" selected="true"/>
      <package id="S3NVESCX_201005251632080655" name="Generic Data
Collection Query Support" selected="true"/>
    </group>
    <group display-name="Account Tracking" expanded="false">
      <package id="WUHJYFNL_201003011427170743" name="Account Tracking
Common" selected="true"/>
      <package id="MMXLVRGT_201003011554580470" name="Active Directory
Account Tracking" selected="true"/>
    </group>
  </optional>
</features>
```

Package Constraints

The package constraints list the restrictions associated with a package. These options can only be changed when a package is being developed. After a package is released or imported, these items cannot change.

Table 3-21 *Package Constraints Settings*

Constraint	Description
IDM Compatibility	Lists the minimum and maximum versions of OpenText Identity Manager that the package supports. These settings are always populated.
Application Compatibility	Lists the minimum and maximum versions of the application the package supports. These settings are not required for all packages.
Driver Type	Lists all of the supported driver types the package can be used with.

Package Dependencies

The Package Dependencies property page list the packages that the current package needs to run. Packages are divided up into much smaller pieces than a driver configuration file. Some packages have dependencies on other packages and some do not.

Table 3-22 *Package Dependencies Settings*

Setting	Description
Name	Lists the name of the package that is a dependency.
Minimum	Lists the minimum version of the package dependency.
Less than	Lists the highest version of the package dependency.
Exceptions	If there is a version of the package that is not a dependency, it is listed as an exception.
Add dependency	Allows you to add dependencies to the package you are currently developing. This option is not available for released packages.
Remove dependency	Allows you to remove dependencies to the package you are currently developing. This option is not available for released packages.

Package Initial Settings

The initial settings are used by package developers to create a template of items that are required for a driver to start. This information is specified in ds-object code that modifies the driver object at installation. The ds-object code installs driver shim parameters, driver start options, named passwords, GCVs, and filters. Unlike other package content, these settings cannot be uninstalled.

Package Languages

The Package Languages property page lists the languages that package is translated into.

Package License

The Package License property page lists the license for the package.

Package Linkage

The Package Linkage property page lists all of the places the package is linked to in your project. Linking allows you to install content in package A and link to this content in package B. This allows you to create generic policies that can be reused, then link the policies with minor differences for a specific driver.

NOTE: The **Package After or Before Linkage** option of Package Manager might not work as expected. To confirm if you are using these package linkages, run the Project Checker. As an alternative, we recommend that you use the Package Weights option of Package Manager to order the policies in a policy set.

Package Readme

The package Readme lists the information the developer wants you to know about the package. For example, it can contain a list of new features in a package version, what the linkage directives should be for a package, and a change log for the package. For more information about package development, see Section 7, “Developing Packages,” on page 193.

Package Targets

The package targets are all of the places where the package is installed in your project. This allows you to see where the package is being used if you need to uninstall a package.

Package Vendor

The package vendor information is listed on this property page. This allows you to contact the vendor of a package if you need more information about a package.

Table 3-23 *Vendor Settings*

Setting	Description
Vendor Name	Specify the vendor name. If this is for internal consumption, specify the name of your company.
Vendor Address	Specify the address for the vendor or your company.
Vendor URL	Specify the URL of the vendor or your company.
Vendor eMail	Specify an e-mail for the vendor or your company.
Contact Name	If there is a specific contact person for this package, specify his or her name.
Contact eMail	If there is a specific e-mail address for the contact person, specify it in this field.

Configuring Package Content

You can view or change configuration settings for the content of a package. You can change the content only when the package developer mode is enabled on the Identity Vault. For more information, see Section 7, “Developing Packages,” on page 193.

To view the properties of the package content, expand any package, then right-click the content and click **Properties**.

- “Package Content General Settings” on page 116
- “Package Content Installation” on page 116
- “Package Content Linkage” on page 116

Package Content General Settings

You can either view or change the general settings for the package content.

Field	Description
Name	Displays the name of the item in the package.
Notes	Displays any notes about the content of the package.

Package Content Installation

This page displays the installation directive for the package content. It lists the order of installation of the content in the package. If you have multiple policies, it lists the order that the policies are executed.

Package Content Linkage

This page displays the order of how the policy is linked in the policy set. This displays the order that the policies are executed in the policy set even if the policies are part of separate packages.

Configuring Prompts

Prompts are Global Configuration objects that are contained in packages. The prompts are the fields that are presented to users when they create a driver. The prompts are created by developers so users can configure the driver correctly. For more information, see “Adding Default Package Prompts” on page 222.

Prompts are stored in a Resources folder under the package in the package catalog. To see the properties of the prompt, right-click the prompt, then click **Properties**.

- “Prompts General Settings” on page 117
- “Prompts” on page 117
- “Prompts Transformation” on page 117
- “Target Transformation” on page 118

Prompts General Settings

You can change many of the general settings for the prompts.

Table 3-24 *Prompts General Settings*

Setting	Description
Name	Displays the name of the prompt. You cannot change the name of the prompt. It is set when the prompt is created. The name of the prompt is a combination of the package name and the prompt type.
Type	A list of the different prompt types. You can change the prompt type. The prompt types are: ◆ Driver Name ◆ Global Configuration ◆ Initial Settings ◆ Job ◆ Remote Loader ◆ Upgrade Settings ◆ MSysInfo Classification ◆ Custom
Order	This is the order in which the prompts are displayed when a driver is configured. 0 is the first prompt that is displayed and the rest are in ascending order.
Targets	Click **Add** or **Remove** to add and remove the packages the prompt is part of. The package you created the prompt on is the first package listed.

Prompts

The **Prompts** field is an example of what is displayed when the package is configured. You can validate that the prompts are displayed properly before configuring a package.

Prompts Transformation

Displays the transformation style sheet for the prompt resources GCV document, based on the GCVs of other prompts that appear before this prompt in the sorted package prompt list. This style sheet is created by default when the prompt is created. You can modify the style sheet on this page.

If you have made changes to the style sheet, you can clear the changes and revert to the default style sheet:

1. Click **Generate from template**.
2. Select the template type, then click **OK**.

Target Transformation

Displays a transformation style sheet that allows the prompts to modify the package items in the targets of the prompts. You can modify the style sheet on this page.

If you have made changes to the style sheet, you can clear the changes and revert to the default style sheet:

1. Click **Generate from template**.
2. Select the template type, then click **OK**.

Configuring Global Configuration Objects

Global Configuration objects contain global configuration variables (GCVs) and are used when the configuration values are referenced from content in packages.

- "Global Configuration Object General Settings" on page 118
- "Global Configuration Object GCVs" on page 118

Global Configuration Object General Settings

The General Settings page allows you to change the name of the Global Configuration object.

Global Configuration Object GCVs

The GCVs page displays the GCVs that are contained in the Global Configuration object. You can add, edit, and remove the GCVs through this page. You can also edit the GCVs in XML instead of using the editors provided.

Configuring Jobs

Designer has a job scheduling utility to schedule events. Through this utility, the system can be set to disable an account on a specific day, or to initiate a workflow to request an extension for a person's access to a corporate resource. Designer's job scheduler contains the same functionality as the job scheduler found in iManager. For information about creating jobs, see "Creating a Job" on page 376.

In the Outline view, right-click the Job icon, then select Properties.

- "General" on page 118
- "Trace" on page 119

General

You have one selection under the General heading: **Policy Name**. You can change the job's name by modifying the name that appears in the **Policy Name** entry, then clicking **OK**.

Trace

Through the Modeler, you can add a trace level to your jobs. With the trace level set, DS Trace displays the OpenText Identity Manager events as the engine processes the events. The trace level only affects the driver where it is set.

IMPORTANT: You should use the trace level only for testing or for troubleshooting driver issues. Setting a driver trace level on a production driver can cause OpenText Identity Manager server to process events slowly.

Table 3-25 *Job Trace Settings*

Field	Description
Trace level	As the job trace level increases, the amount of information displayed in DS Trace increases. Trace level 1 shows errors, but not the cause of the errors. To see password synchronization information, set the trace level to 5.
Trace file	Specify a filename and location where the OpenText Identity Manager information is written for the selected driver. When a value is set in this field, all Java information for the job is written to file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
Trace File Encoding	The trace file uses the system's default encoding. You can specify another encoding if desired.
Trace file size limit	Allows you to set a limit for the Java trace file. If you set the file size to Unlimited, the file grows in size until no disk space is available. **NOTE:** The trace file is created in multiple files. OpenText Identity Manager automatically divides the maximum file size by ten and creates ten separate files. The combined size of these files equals the maximum trace file size.
Trace name	Helps you track job trace messages. The name that you specify here appears with the job trace messages.

For more information about viewing as saving trace information with DS Trace, see "Driver Trace Levels" on page 107.

Configuring ID Policy Containers

An ID Policy container is a repository for ID policies and is used in conjunction with the ID Provider driver. For more information about the ID Provider driver, see the Driver for ID Provider Implementation Guide (https://www.netiq.com/documentation/identity-manager-4.10-drivers/

idprovider/data/netiq-identity-manager-driver-for-id-provider.html). When the ID Provider driver receives an ID request from a client, it generates an identification that is based on the ID policy specified in the request and passes the identification to the client.

To configure an ID Policy container, you must first add the ID Provider driver to a driver set that accesses an Identity Vault. Then, under the ID Provider driver, create an ID Policy container by right-clicking the ID Provider driver and selecting **New > ID Policy Container**. After the container is created, double-click the ID Policy container in the Outline view, or right-click the ID Policy container and select **Properties**.

Table 3-26 *ID Policy Container General Settings*

Field	Description
Name	The name of the ID Policy container. You can change the name as necessary.
Notes	You can add notes to better define how you are using the ID Policy container.

In order for ID policies to work, you must also add and configure an ID policy in the ID Policy container. See "Configuring ID Policies" on page 120.

Configuring ID Policies

An ID policy allows the ID Provider driver to generate unique IDs. When the ID Provider driver receives an ID request from a client, it generates an identification that is based on the ID policy specified in the request and passes it to the client.

The ID Provider driver can act as a client itself and can assign IDs to objects in the Identity Vault. For more information about the ID Provider driver and its components, see the Driver for ID Provider Implementation Guide (https://www.netiq.com/documentation/identity-manager-4.10-drivers/idprovider/data/netiq-identity-manager-driver-for-id-provider.html).

To configure an ID policy, you must first add the ID Provider driver to a driver set. Then, under the ID Provider driver, create an ID Policy container and add an ID policy. After the ID policy is created, double-click the ID policy in the Outline view, or right-click the ID policy and select **Properties**.

Figure 3-5 *ID Policy General Properties Page*

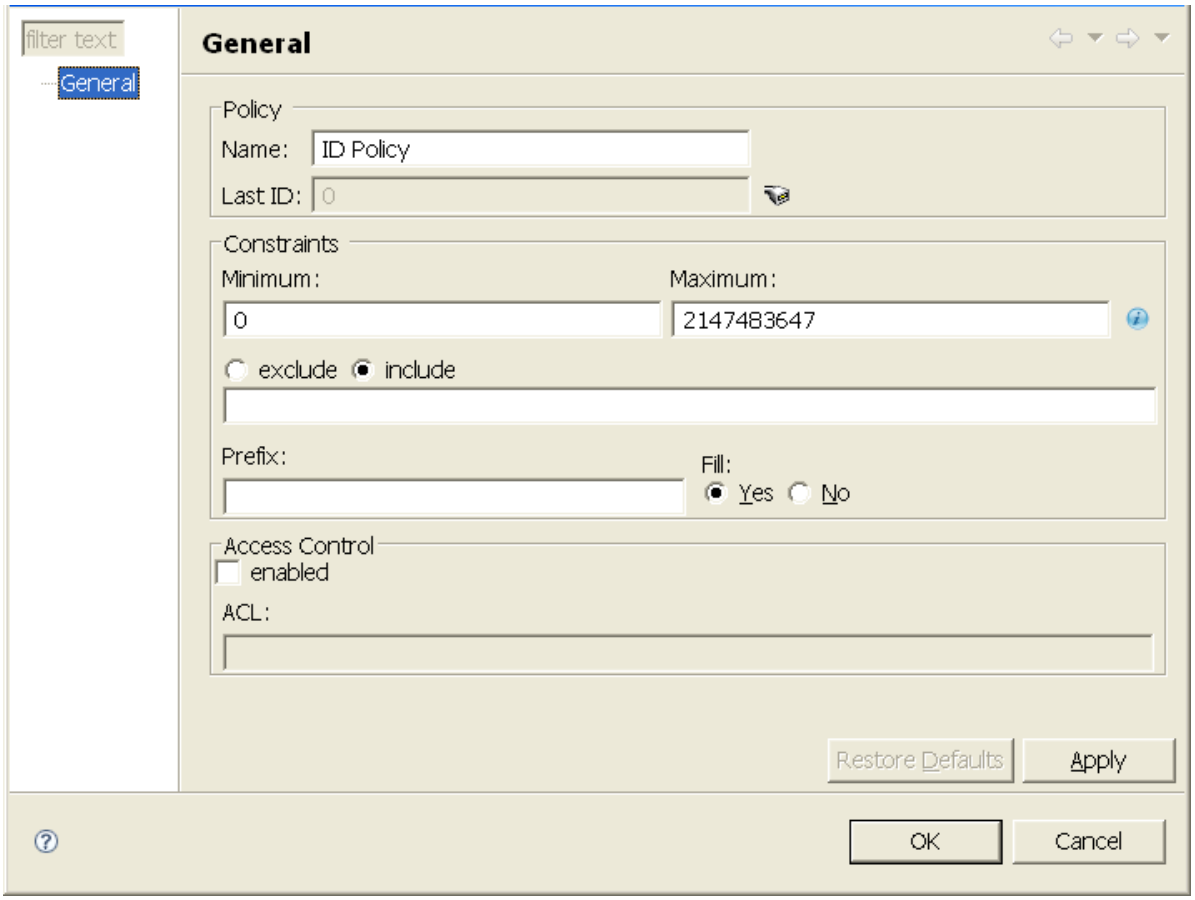


Table 3-27 *The ID Policy General Settings*

Field	Description
Policy Name	The name of the ID policy.
Policy's Last ID	The last ID number that was used by this ID policy. If you have deployed this ID policy, use the **Connect** icon to update this field to the last ID number that was stored in the Identity Vault for this ID policy. **NOTE:** Only the ID Provider driver can update the last value stored in the Identity Vault.
Constraints Minimum/ Maximum	Numbers must be between 0 and 2147483647. If you have a fixed system that can only handle eight digits, set the **Maximum** to 99999999.
Constraints Exclude/ Include	Allows you to include or exclude a set of numbers that you type. Numbers can be typed in a comma-delimited list and you can use ranges, such as 10,100,1000,5000-10000,1099, etc.
Constraints Prefix:	Allows you to give a prefix to the IDs that are generated using this ID policy. If you create multiple ID policies, a prefix is useful to see which ID policies are being used. An example is WFID, for workforce IDs.

Field	Description
Constraints Fill: Yes/No	If you choose Yes, the ID is filled with leading zeros (0) up to the maximum length. This helps keep generated IDs at the same length. If you select No, it does nothing and the ID lengths increment over time.
Access Control Enabled	Check this box if you want to enable access control lists.
Access Control ACL:	Type the names of the access control lists you want to use. Access control must be enabled before you can type in ACLs.

Configuring a Notification Template

You can use the property page for a Notification Template to change the name of the notification template.

1. In the Outline view, expand **Default Notification Collection**.
2. Right-click a notification template (for example, Forgot Password), then select **Properties**.
3. Edit the name, then click **OK**.

For additional configuration information about notification templates, see Chapter 10, “Setting Up E-Mail Notification Templates,” on page 271.

Configuring Application Properties

To view or change an application’s settings, double-click the application (for example, LDAP Directory) in the Modeler.

- “General” on page 122
- “AD Domain” on page 123
- “Administrator” on page 123
- “Connectivity” on page 123
- “Environment” on page 127

General

Table 3-28 *Application General Settings*

Field	Description
Type	Changes the type of application your driver connects to. For example, if you configure a JDBC driver to connect to a MySQL* database, but then need to change to an Oracle database, you can scroll to **Database**, select **Oracle**, then click **Apply**.
Browse	Enables you to navigate to and select an image file.

Field	Description
Name	Enables you to customize the application's name or label.
Version	Enables you to document the application's version.

AD Domain

You can capture information about an Active Directory application. This information is useful if you want Document Generator to include this information when you document the project.

If you provided information in the LDAP settings, Designer populates the AD Domain fields.

Administrator

The **Administrator** option is divided into three sections. Entering information in these sections is optional.

- **Personal Information:** Use this section to enter information specific to the Identity Vault, such as Name, Title, Department, and Location,
- **Contact Information:** Use this section to enter information such as Email, Phone, Cell phone, Pager, and Fax.
- **Notes:** Use this section to type any reminders you might need for future reference.

Connectivity

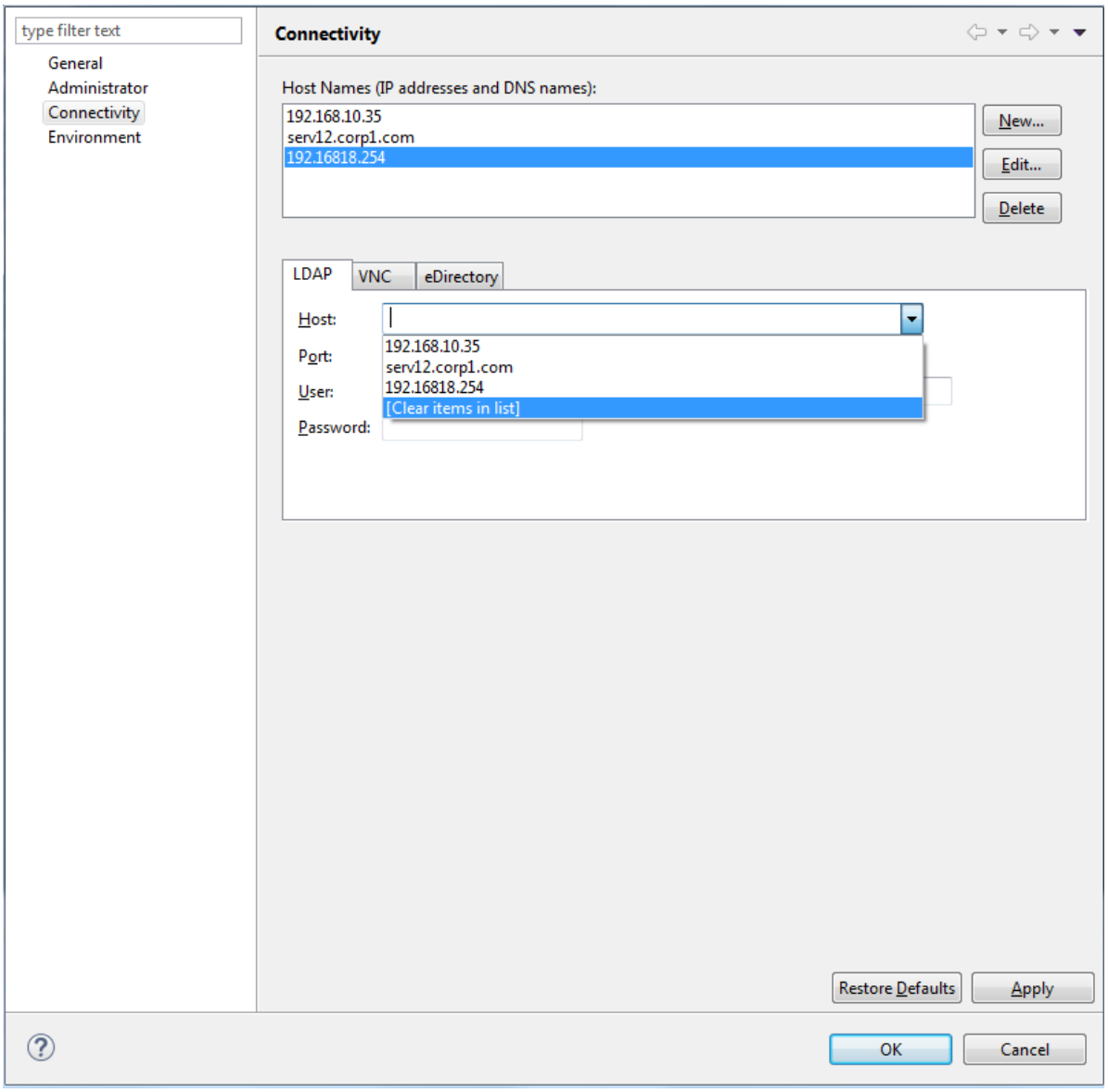
- "Host Names" on page 123
- "LDAP" on page 125
- "VNC" on page 125
- "OpenText eDirectory" on page 126
- "Configuring a Remote Connection" on page 126
- "Customizing the Viewer" on page 126

Host Names

NOTE: This control is available only for OpenText eDirectory applications.

The **Host Names** field lets you create a list of server IP addresses and DNS names for your OpenText eDirectory application. Because servers can have multiple IP addresses and DNS names, it is useful to be able to create a list of those host names that you can easily access when configuring connectivity for your OpenText eDirectory application.

Figure 3-6 *Host Name List for OpenText eDirectory Applications*



You can add, modify, and delete host names from the Host Names list.

When you specify a host on the **LDAP**, **VNC**, or **eDirectory** tabs, the host entry is automatically added to the Host Names list.

Double-click an entry in the Host Names list to automatically populate the **Host** field in the **LDAP**, **VNC**, or **eDirectory** tabs.

Host entries in the Host Names list are also available from the **Host** field drop-down list in the **LDAP**, **VNC**, and **eDirectory** tabs.

LDAP

You can configure some applications (for example, Active Directory, OpenText eDirectory, and LDAP) for an LDAP connection. If the application doesn't support an LDAP connection, the **LDAP** tab doesn't display.

Host: The server's IP address or DN.

Port: The server port to communicate with the directory.

User: The user's name (in LDAP format).

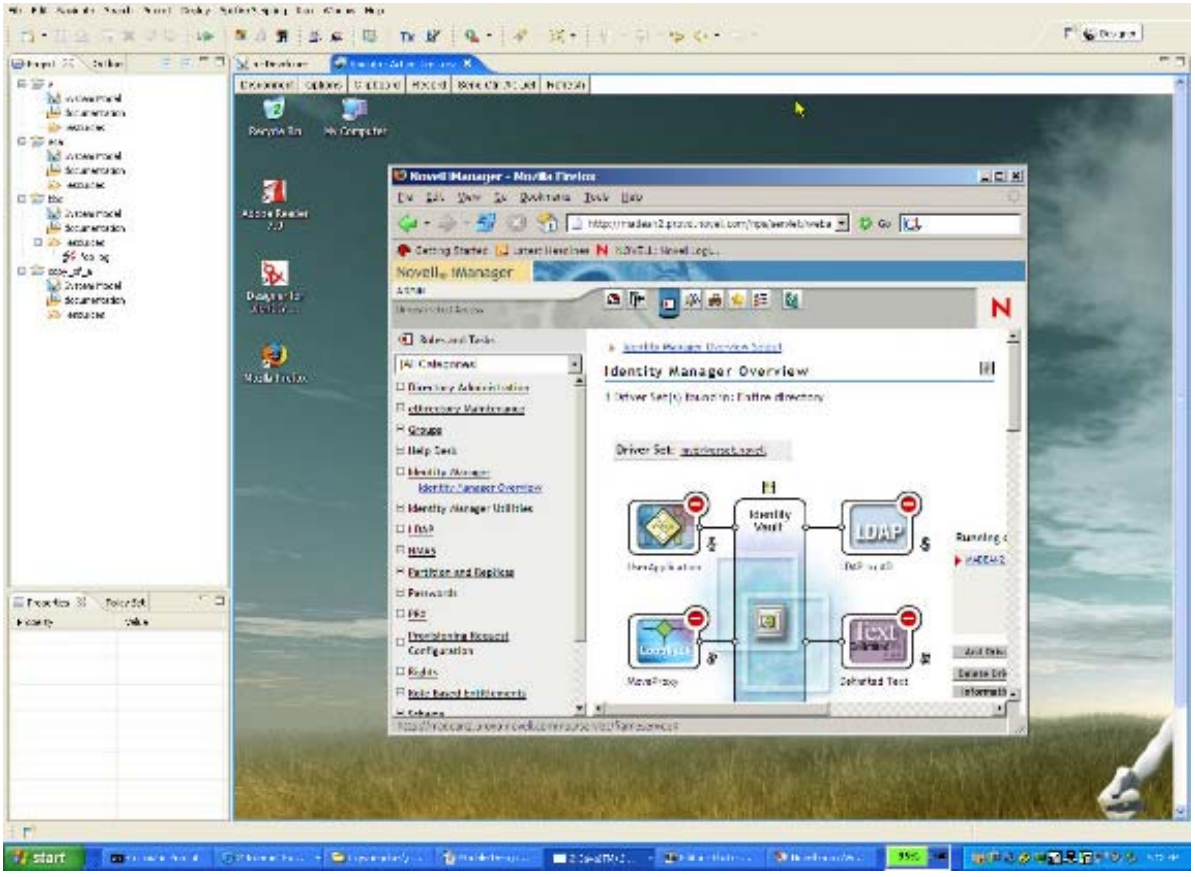
Password: The user's password.

VNC

From within Designer, you can view the desktop of the machine that is running your applications, and remotely control that desktop by interacting with it. This feature enables you to administer users or your applications with the native tools of that system, from one location.

This functionality is hosted in an embedded editor inside Designer. You can have multiple remote control sessions with different systems, all open at the same time.

Figure 3-7 *A Remote Desktop*



OpenText eDirectory

You can configure connectivity to OpenText eDirectory applications. This is similar to configuring an LDAP connection, but uses native OpenText eDirectory protocols instead of LDAP.

Host: The server's IP address or DN.

Port: The server port to communicate with the directory.

User: The user's name (in OpenText eDirectory format).

Password: The user's password.

Configuring a Remote Connection

To remotely control a desktop, the machine that is running your application needs to have a VNC (virtual network computing) server installed and running. You can usually download a free VNC server from the Internet.

You can easily configure any system or design element in Designer for this feature by editing any application or design element:

1. Right-click an application or design element.
2. Select **Properties** > **Connectivity**.
3. On the **VNC** tab, type the authentication information.

   Host: The DN (for example, server33.houston.company.com) of the server where the VNC is running.

   Port: Typically 5901 for Linux servers or 5900 for Windows.

   Password: The password to the VNC server.
4. Click **OK**.

Customizing the Viewer

A toolbar at the top of the desktop viewer enables you to configure the following:

- Encoding type (RAW, RRE, CoRRE, Hextile, Zlib, Tight). The default is Tight.
- Compression level
- JPEG Image Quality (0 - 9). The default is 6.
- Cursor shape updates. The default is Enable.
- Use CopyRect. The default is Yes.
- Mouse buttons 2 and 3. The default is Normal.
- View only. The default is No, so that you can interact with the desktop.
- Clipboard
- Record session and save to file.
- Send Ctrl+Alt+Delete.
- Refresh

For more information, see the TightVNC documentation Website (http://www.tightvnc.com/).

Environment

You can enter notes about the application's platform, hardware, and environment.

Adding Prompts to a Driver Configuration File

Several node types are defined for driver configuration files. These extensions were made to support the following:

- Prompting once for a value that is used repeatedly throughout a single driver configuration file.
- Prompting once for a value that is used across multiple driver configuration files, as part of the Import Drivers Wizard.
- Allowing the user to select a value from a drop-down list of values.
- Global modification of the driver configuration file according to a contained XSL style sheet.
- Built-in variables that can be referenced without declaring them, in order to access information about the driver and its environment (a tree name, driver set name, driver set DN, server name, server DN, driver name and driver DN).
- The ability to "layer" prompts. It is possible to ask the user multiple sets of questions, with the second and later sets being controlled by the user's responses to prior sets. For more information, refer to "Editing Driver Configuration Files" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

The primary new node types are variable-decl, variable-ref, and xsl-modify.

Table 3-29 *New Node Types*

New Node Type	Description
variable-decl	Allows you to define driver configuration variables that are prompted for (optionally) and replaced into a driver configuration file during its import. Multiple variable-decl blocks can be used to define a "layered" set of prompts. Refer to "Editing Driver Configuration Files" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.
variable-ref	Used to reference a variable defined in a variable-decl within your driver configuration files.
xsl-modify	Used to globally modify the driver configuration file after all variables (and prompting) have been resolved. The contents of this node are extracted and used as an XSL style sheet that is applied to the patched driver configuration file.

For information about adding prompts to a sample configuration file, see "Editing Driver Configuration Files" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

Synchronizing Passwords

To view or edit password synchronization, use the Dataflow editor. See "Filtering Views" on page 251 and "Synchronizing Passwords" on page 251.

4 Managing OpenText Identity Manager Versions

Your environment might have different versions of OpenText Identity Manager.

If you configured and wrote policies for an earlier version of OpenText Identity Manager in your environment, you might encounter the following issues:

- You could easily build a solution that would not deploy.
- You did not know which features worked in one environment versus another environment.

To solve these issues, Designer tracks versions of the following objects:

- OpenText Identity Manager engines
- Identity Vaults (trees)
- Drivers

As you use Designer, you see only the user interface of features that apply to the version that you are working on. Project Checker and Deploy ensure that what you have configured is supported in the target environment.

- "Key Differences in OpenText Identity Manager Versions" on page 130
- "Changing the OpenText Identity Manager Version" on page 130
- "Tracking Versions of OpenText Identity Manager" on page 131
- "Support for Driver Configuration Versions" on page 132
- "Checking Projects for Version Issues" on page 134
- "Adjusting the UI Based on the Version Number" on page 134

Key Differences in OpenText Identity Manager Versions

Identity Manager 3.5	Identity Manager 3.6	Identity Manager 4.7
◆ New object types were added: ◆ ECMAScript Objects ◆ Jobs ◆ Mapping Table Resource Objects ◆ Resource Libraries ◆ New Policy Linking capabilities where a policy can be in multiple lists ◆ Many new DirXML Script actions, conditions, tokens, and verbs ◆ Ability for DirXML Script to nest conditions ◆ Driver-scoped local variables in DirXML Script that let you refer to variables outside of the policy	◆ Support for 64-Bit operating systems ◆ New installation program ◆ New driver configuration files ◆ Driver health monitoring ◆ New ID Provider driver ◆ Reciprocal Attribute Mapping ◆ Additional DirXML Script elements ◆ Nested group support ◆ User Application	◆ Integrated installer ◆ Packages ◆ Installation ◆ Management ◆ New Resource Objects ◆ Global configuration resource objects ◆ Package prompt resource objects ◆ DS resource objects ◆ SharePoint driver ◆ Salesforce.com driver ◆ Identity Reporting Module

Changing the OpenText Identity Manager Version

You can import and deploy to all versions of OpenText Identity Manager from version 4.0.2 or later.

When you convert an earlier project, Designer defaults the OpenText Identity Manager version numbers to the latest version. During conversion, Designer informs you that this default is being applied.

You can change this version number by doing either of the following:

- In the Outline view, right-click the Server object, select **Properties**, then select from the **Identity Manager Version** drop-down list.
- In the Modeler, select an Identity Vault, click **Window** > **Preferences**, expand **NetIQ** and select **Identity Manager**, then select a version from the drop-down list.

You can also find information about upgrades, downgrades, and a link to a help topic. This information explains the key differences between versions of OpenText Identity Manager.

When you import into a new server (or create a server based on a server that you have browsed to in the directory), the new server inherits the imported version of OpenText Identity Manager.

If you do a live update in the server properties page, Designer updates the server to the current version of OpenText Identity Manager that is in the target environment.

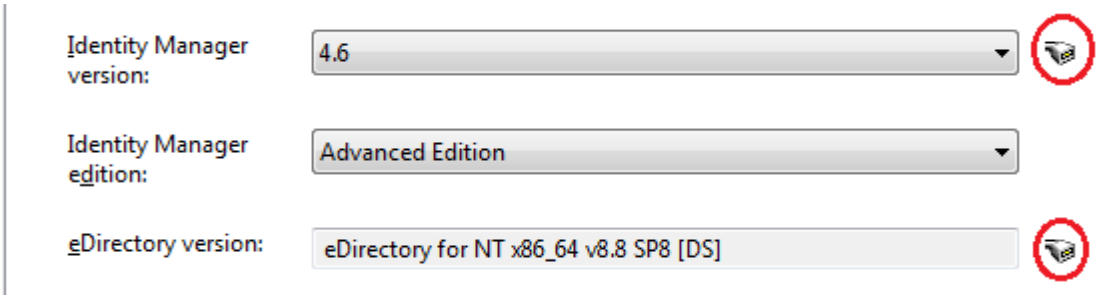
Tracking Versions of OpenText Identity Manager

Designer tracks the OpenText Identity Manager version. Filtering functionality is based on this version information. When multiple servers are associated to a driver set, Designer calculates an “effective engine version.” This version is the earliest OpenText Identity Manager version in the driver set.

If you want to use the latest Identity Manager 4.7 features, it is important that all servers belonging to the driver set are upgraded to 4.7. This version can be manually upgraded or downgraded from the server properties page.

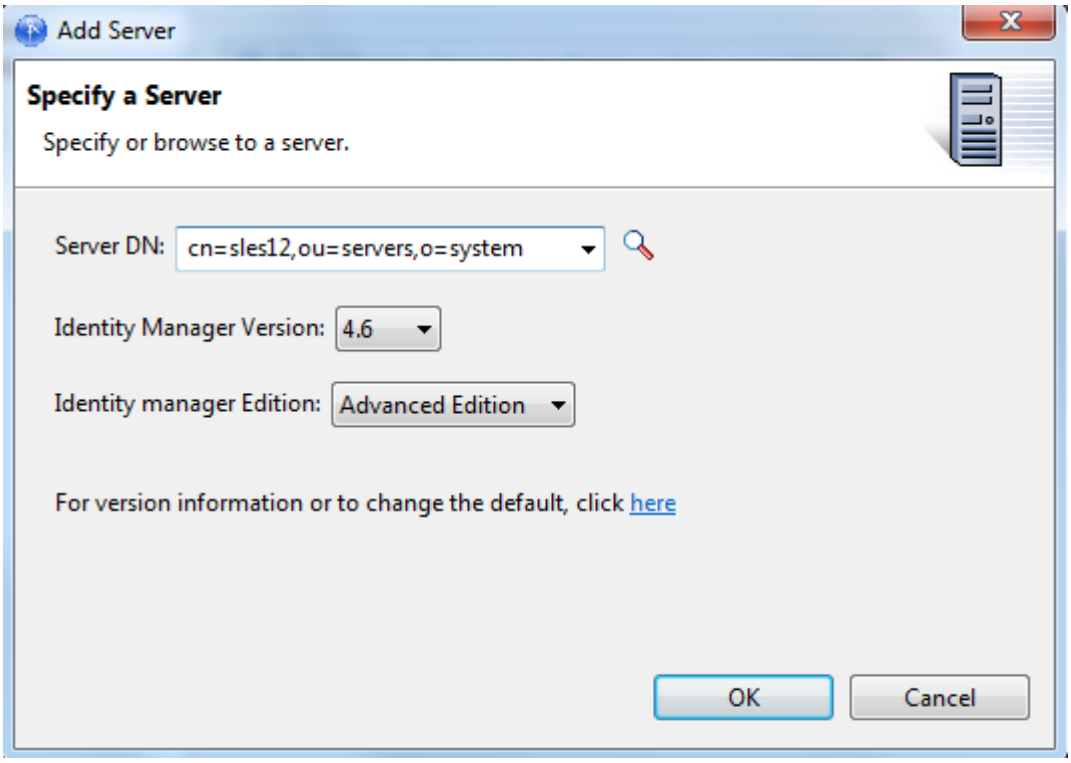
Additionally, live update icons retrieve current OpenText Identity Manager and OpenText eDirectory version information about the server properties page.

Figure 4-1 *Live Update icons*



The **Add Server** dialog box allows you to specify version information when an Identity Vault is created.

Figure 4-2 *The Add Server Dialog Box*



The Driver Set Log Level and Driver Log Level property pages have dynamic version widgets next to any log event that is not supported by your effective OpenText Identity Manager version. Designer displays a warning message for an unsupported log event.

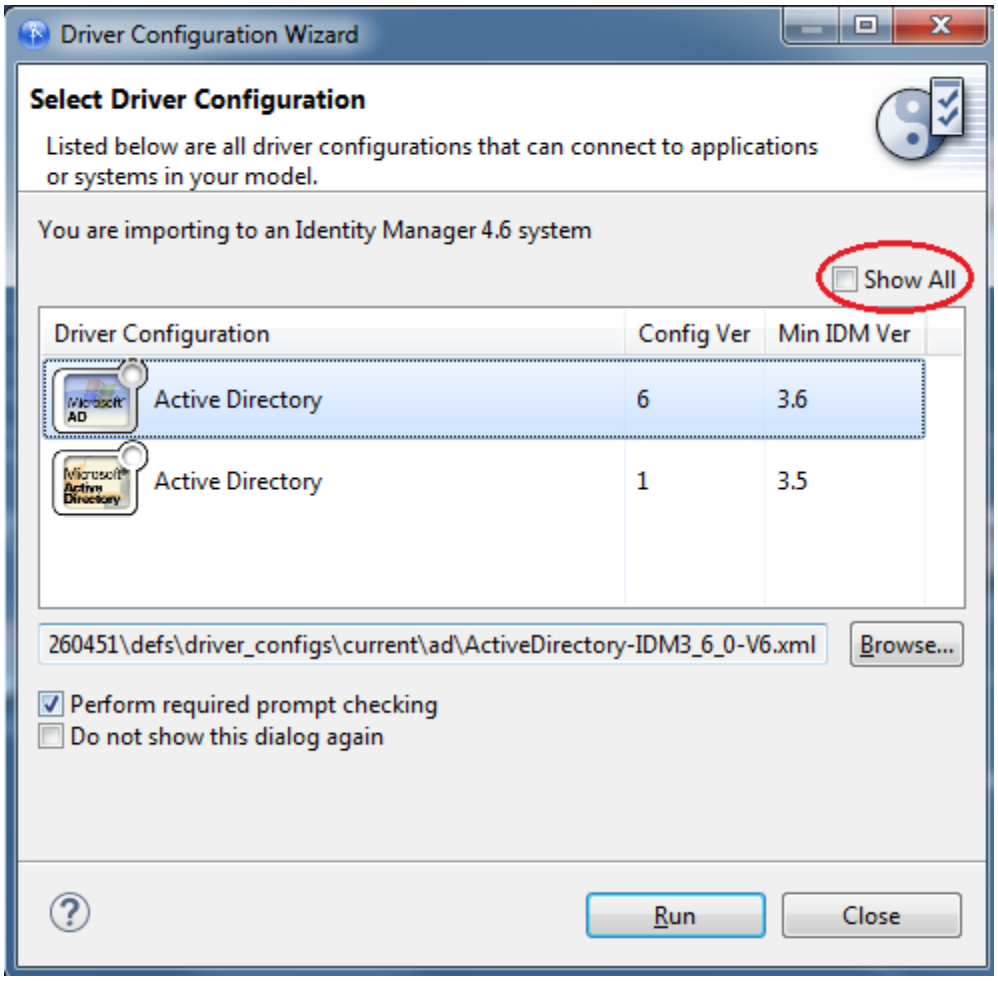
Support for Driver Configuration Versions

Starting Identity Manager 4.0, driver configuration files are replaced with packages. You can still use driver configuration files. However, new and updated content for drivers is contained only in packages.

The Driver Configuration Wizard provides the following versioning information about the driver configuration files and your OpenText Identity Manager solution.

- The engine version that you are importing into. This information is taken from the current project. You control the version number.
- A descriptive name of the driver configuration.
- The version of the configuration as a single (undelimited) version number.
- The minimum required engine version for this configuration to run.
- The full filename of the selected list item. This name is below the list. It is displayed there for transparency.
- A check box that indicates possible unrecommended or incompatible configuration files.

Figure 4-3 *A Deselected Show All Check Box*



By default, the **Show All** check box is deselected if unrecommended or possibly incompatible configuration files are available. If all available driver configuration files are recommended and guaranteed compatible, the check box is dimmed and selected, indicating that all available options are displayed.

A deselected **Show All** check box implies the following:

- Additional driver configuration files are available but they are not recommended.
- The additional driver configuration files are probably incompatible with the engine version that you are importing to.

When you select the **Show All** check box, the list contains many more items. These new items were previously hidden because the minimum required engine version for them is 3.5. Because the user is importing to 4.7, the configuration might be incompatible.

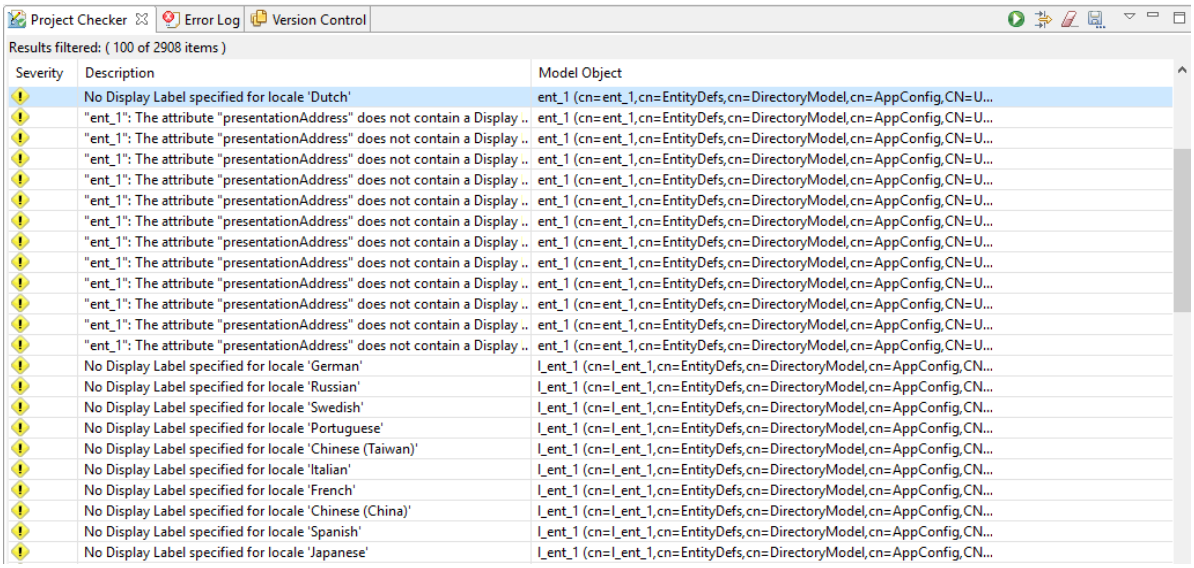
Checking Projects for Version Issues

A full suite of project checks makes sure that what you have configured makes sense for your target environment and can be successfully deployed. Designer's UI blocks the creation of unsupported objects and hides features based on the version number. Nevertheless, unsupported actions might still occur through a few “back-door” methods, such as copying and pasting, importing, and downgrading your server after you have configured for a newer environment.

In all of these instances, Project Checker catches the problems.

For example, for policy libraries to work, all of the servers on a given driver set need to be at the same OpenText Identity Manager version. Project Checker catches problems like this where you might have an unsupported mix of servers. In this case, the project check results would look like the following figure:

Figure 4-4 *Project Checker*



Version problems are sorted to the top and have a version icon. If you double-click the item, you get more details about the problem and how to resolve it.

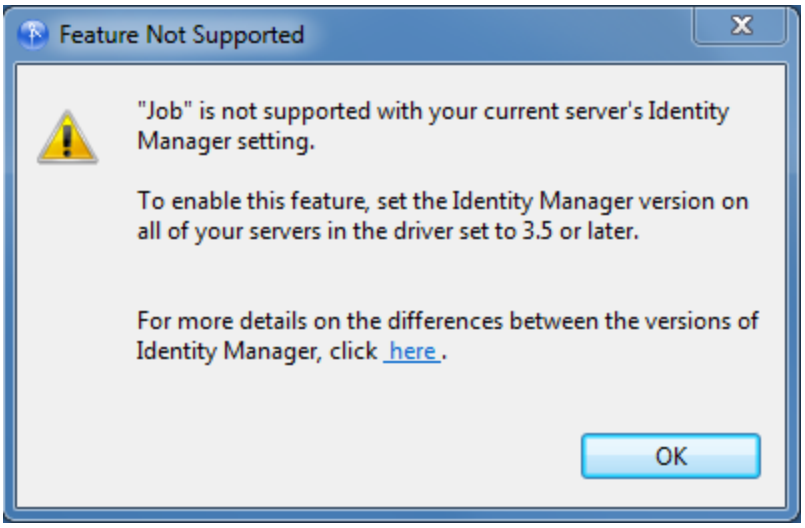
Adjusting the UI Based on the Version Number

Designer displays and enables or disables capabilities based on the version of OpenText Identity Manager that is associated with the engine. For example, if you edit a policy that is associated with a server that uses a specific version of OpenText Identity Manager, Policy Builder shows you all of the new actions, conditions, verbs, and tokens that ship with that release. This feature lets you try out the next version of OpenText Identity Manager before it is even released.

Also, if you set the server to the latest version of OpenText Identity Manager, you get the previous version of Policy Builder that Designer has shipped with in the past.

If you try to create an object that is not supported by your server version, a prompt tells you that this action isn't supported. For example, Identity Manager 3.5 introduces the concept of Jobs, Mapping Tables, and Policy Libraries. If you try to create one of these objects on a 3.0.1 server, you see the following message:

Figure 4-5 *Prompt: Feature Not Supported*



Future milestones of Designer will continue to evolve the UI to better handle version differences.

5 Managing the Schema

Designer includes a copy of the base Identity Vault schema, which is stored in the `BaseIVSchema.xml` file. This file is located in `\Designer\plugins\com.novell.core.datatools_x.x.x.x\defs\schema`, where *x.x.x.x* represents the specific Designer build.

Do not directly modify `BaseIVSchema.xml`. Instead, use Designer to add the schema information from this file into your project. The Manage Schema tool allows you to change the schema as part of the project without modifying the original `BaseIVSchema.xml` file.

You can add, delete, rename, and modify classes and attributes in the Identity Vault schema. You can import the Identity Vault schema from the production environment, or use the default schema. After modifying the schema, you can deploy it into the production Identity Vault.

WARNING:

- If you do not have a good understanding of how the Identity Vault schema works, changing the default schema can cause data corruption.
- If you create new classes and attributes, do not use spaces in these object names.
- If you modify classes or attributes and then deploy the modified schema into a tree where these classes are in use, one of the following problems can occur:
  - Those objects can become unknown.
  - Synchronization errors can occur.

To understand the basics of the schema, see "Managing the Schema" (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/a4a9bz0.html) in the *OpenText™ eDirectory CE 24.4 (v9.3) eDirectory Administration Guide (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/bookinfo.html)*.

If you subscribe to LogicSource, see *eDirectory Best Practices Guide in LogicSource* (http://support.novell.com/subscriptions/articles/novell_logicsource.html) for additional information. LogicSource is a subscription-based service that OpenText provides to its customers.

- "Using the Manage Schema Tool" on page 138
- "Creating Classes and Attributes" on page 148
- "Modifying the Schema" on page 150
- "Deploying the Schema into the Identity Vault" on page 151
- "Exporting the Schema to a File" on page 154
- "Importing the Schema" on page 158
- "Managing a Copy of an Application Schema" on page 166
- "Mapping Identity Vault to an LDAP Schema" on page 167
- "Comparing the Schema" on page 167

Using the Manage Schema Tool

To open the Manage Schema tool, right-click an Identity Vault object in the Modeler or Outline View, then select **Manage Vault Schema**.

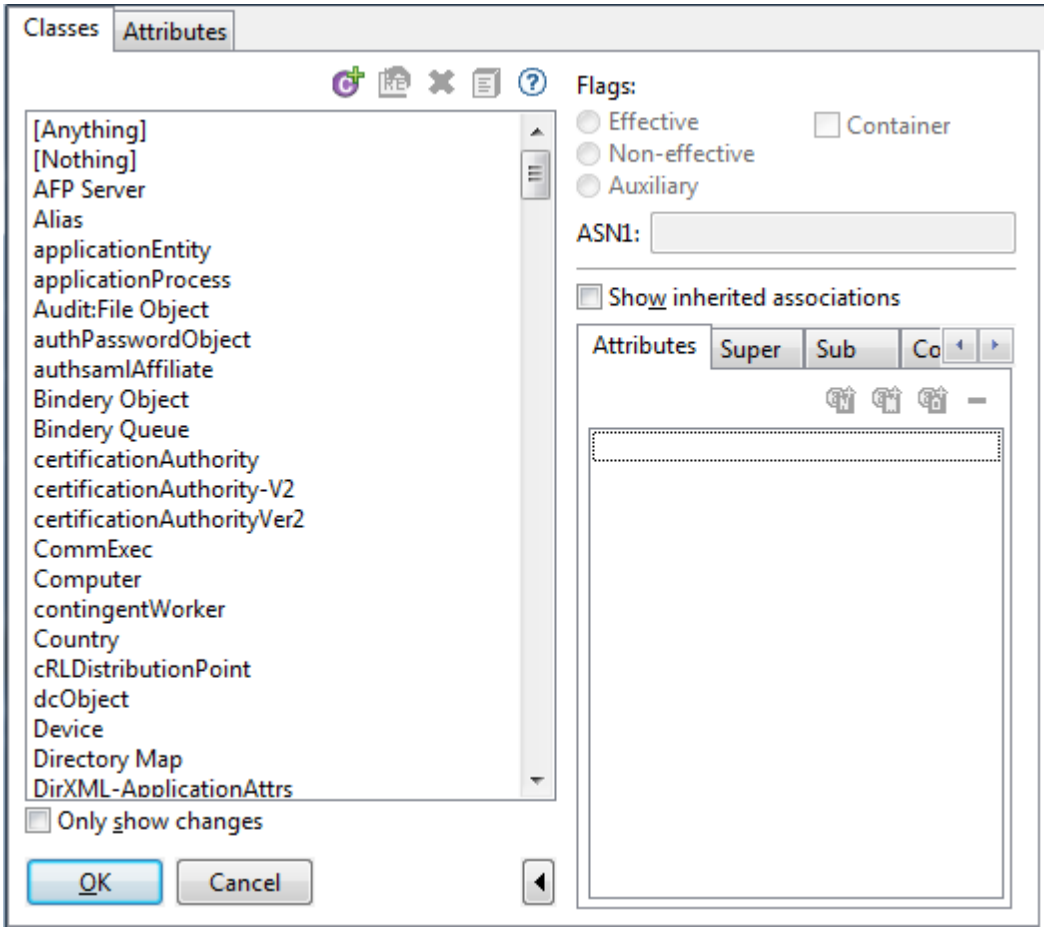
If a custom schema in the production environment needs to be tested, you can import the schema into Designer. After you have tested and modified the schema, you can deploy it into the production environment. For information about importing schema, see "Importing the Schema" on page 158.

The Manage Schema tool lets you add, delete, rename, and modify classes and attributes in the Identity Vault schema. The class information and the attribute information is organized into separate tabs in the Manage Schema tool.

- "The Classes Tab" on page 138
- "The Attributes Tab" on page 142

The Classes Tab

From the **Classes** tab, the Manage Schema tool lets you add, delete, rename, and modify schema classes.



The **Classes** tab includes the following components:

- "Class List Toolbar" on page 139
- "Only Show Changes" on page 139
- "ASN1" on page 139
- "Flags" on page 140
- "Show Inherited Associations" on page 140
- "Associations List" on page 140

Class List Toolbar

The Classes list includes the following tools:

Table 5-1 *Classes List Toolbar*

Icon	Description
Add Class 	Launches the New Class Wizard to create a new Identity Vault class.
Rename Class 	Renames any non-base class. You cannot rename base classes.
Delete Class 	Deletes any non-base class. You cannot delete base classes.
Schema Notes 	Adds descriptive notes to any non-base class.You cannot add notes to base classes.

Only Show Changes

The **Only show changes** check box is below the Classes list. When it is selected, the Classes list displays only those classes that are not part of the base schema, as defined in `BaseIVSchema.xml`. If no non-base classes exist, the Classes list is empty.

Deselect **Only show changes** to see a complete list of base and non-base classes in the Identity Vault schema.

ASN1

Specifies the class's Abstract Syntax Notation number One ID. The ASN1 ID is important as you plan to make the schema definition publicly available.

If you register your schema definition with OpenText, OpenText assigns your class an ASN1 ID. This unique identifier eliminates the possibility of schema collisions caused by duplicate schema names with different definition structures.

For more information about ASN1, visit the International Telecommunications Union Website (http://www.itu.int/ITU-T/asn1/index.html).

Flags

The Flags options let you modify the class type:

Table 5-2 *Supported Class Types*

Flag	Description
Effective	You can create an instance of the defined object in the Identity Vault.
Noneffective	Only used to define other classes. You cannot create an object of a noneffective class.
Auxiliary	Combines attributes to be added to other classes by extending the object class attribute.
Container	Sets the object to be a container object instead of a leaf object. If it is set to be a container, this object can contain other objects.

Show Inherited Associations

The **Show Inherited Associations** check box determines whether the Associations list displays all attributes associated with a class. When the check box is selected (the default), the Associations list displays both assigned and inherited attributes. When the check box is deselected, the Associations list displays only assigned attributes.

NOTE: When you select **Show Inherited Associations**, you cannot delete entries from the Associations list.

Associations List

The Associations list displays the classes and attributes associated with the selected class. The Associations list includes four tabs, each with a toolbar.

Attributes: The **Attributes** tab displays the attributes associated with the selected class. It also identifies if attributes are mandatory or naming. All unmarked attributes are optional.

The **Attributes** tab includes the following tools:

Class Field	Description
Add Naming	Adds a naming attribute association to the selected class.
Add Mandatory	Adds a mandatory attribute association to the selected class.
Add Optional	Adds an optional attribute association to the selected class.
Delete	Deletes an attribute association from the select class.

Super: The Super tab displays the classes from which the selected class inherits attributes. A class that another class inherits from is called a superclass.

A class can inherit attributes from more than one superclass. The superclass that every class inherits from is Top. No class exists above Top. For example, Group inherits directly from Top, but User inherits from Organizational Person. Organizational Person inherits from Person. Person inherits from ndsLoginProperties, and ndsLoginProperties inherits from Top.

The Super tab includes the following tools:

Class Field	Description
Add Superclass Association	Adds a superclass association to the selected class.
Delete	Deletes a superclass association from the selected class.

Sub: The Sub tab displays all classes that inherit from the selected class. If the Sub tab is empty, no classes inherit from the selected class.

The Sub tab includes the following tools:

Class Field	Description
Add Subclass Association	Adds a subclass association to the selected class.
Delete	Deletes a subclass association from the selected class.

Containment: The Containment tab displays the container classes that can contain the selected class. For example, if you select the Group class, the Manage Schema tool lists the domain, Organization, and Organizational Unit classes, which can contain the Group class.

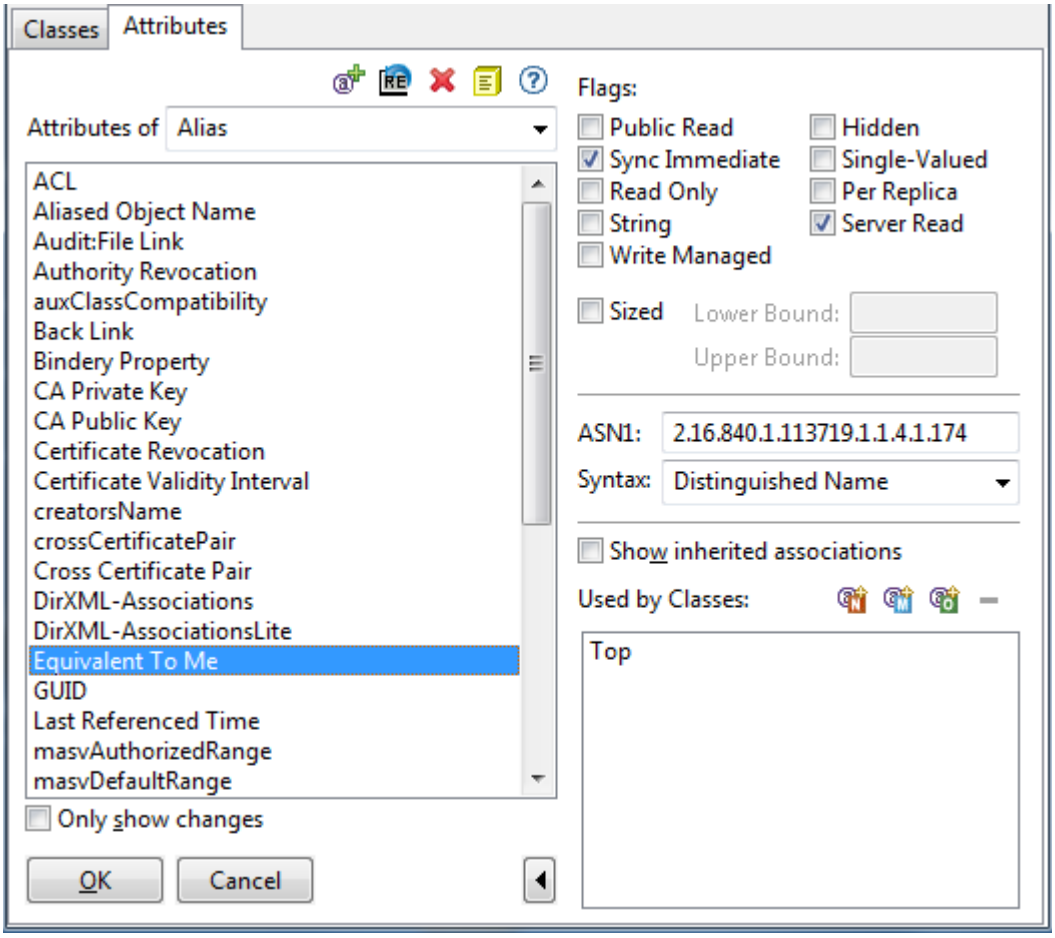
The Containment tab includes the following tools:

Class Field	Description
Add Containment Class Association	Adds a containment class association to the selected class.
Delete	Deletes a containment class association from the selected class.

The Attributes Tab

From the Attributes tab, the Manage Schema tool lets you add, delete, rename, and modify attributes associated with schema classes.

Figure 5-1 *The Attributes Tab on the Manage Schema Toll*



The Attributes tab includes the following components:

- “Attributes List Toolbar” on page 143
- “Only Show Changes” on page 143
- “Flags” on page 143
- “ASN1” on page 144

- “Syntax” on page 144
- “Show Inherited Associations” on page 147
- “Associations List” on page 147

Attributes List Toolbar

The Attributes list includes the following tools:

Icon	Description
Add Attribute	Launches the New Class Wizard to create a new attribute.
Rename Attribute	Renames the selected non-base attribute. You cannot rename base attributes.
Delete Class	Deletes the selected non-base attribute. You cannot delete base attributes.
Schema Notes	Adds descriptive notes to any non-base attribute.You cannot add notes to base classes.

Only Show Changes

The Only show changes check box is below the Attributes list. When this check box is selected, the Attributes list displays only those attributes that are not part of the base schema, as defined in `BaseIVSchema.xml`. If no non-base attributes exist, the Attributes list is empty.

Deselect Only show changes to see a complete list of base and non-base attributes in the Identity Vault schema.

Flags

Attribute flags specify the information that is stored in the attribute and limit the list of acceptable operations that the Identity Vault and OpenText eDirectory clients can perform on the attribute.

Constraint	Description
Public Read	Allows anyone to read this attribute without the read privilege specifically assigned. You can’t use inheritance masks to prevent an object from reading attributes with this constraint.
Sync Immediate	When the attribute is modified, it is synchronized immediately to all of the servers in the replica ring.
Read Only	Only the OpenText eDirectory server process can read this attribute.
String	Allows only string information to be stored in the attribute.
Write Managed	Explicit rights are granted before this attribute can be changed. In order to modify this attribute, users must have managed rights on the object to change the attribute.
Hidden	Only the OpenText eDirectory server process can read this attribute.

Constraint	Description
Single Valued	Allows one value to be stored in the attribute.
Per Replica	Allows one value to be stored in the attribute.
Server Read	The attribute can be read by an NCP server object even though the right to read is not inherited or explicitly granted. The NCP server object is always able to read this attribute, regardless of the rights granted in the ACL.
Sized	Limits the range of values supported by the attribute to some subset of those supported by the attributes data type. For example, you might restrict an Integer attribute to only accept values between 1 and 100.

ASN1

Specifies the attribute's Abstract Syntax Notation number One ID. The ASN1 ID is important is you plan to make the schema definition publicly available.

If you register your schema definition with OpenText, OpenText assigns your attribute an ASN1 ID. This unique identifier eliminates the possibility of schema collisions caused by duplicate schema names with different definition structures.

For more information about ASN1, visit the International Telecommunications Union Website (http://www.itu.int/ITU-T/asn1/index.html).

Syntax

An attribute syntax defines a standard data type that an attribute uses to store its values in the Identity Vault. Each attribute must have a syntax. The following table describes the available syntaxes for Identity Vault attributes.

Syntax	Description
Back Link	The **remoteID** field identifies the backlinked object on the server, and the **objectName** field identifies the server holding an external reference.
Boolean	Two Boolean attributes match for equality if they are both True or both False. True is represented as one (1), and False is represented as zero (0). Any attribute defined by using this syntax is single valued.
Case Exact String	Attributes using this syntax can set size limits. Two Case Exact Strings match for equality when they are of the same length and their corresponding characters are identical.
Case Ignore List	Two Case Ignore Lists match for equality if the number of strings in each is the same, and all corresponding strings match. For two corresponding strings in the list to match, they must be the same length and their corresponding characters must be identical (according to the rules for case ignore strings).
Case Ignore String	Used in attributes whose values are strings and where the case (upper or lower) is ignored.

Syntax	Description
Class Name	Used to match two class names where the case (upper or lower) is ignored.
Counter	The attribute is single valued. The syntax is similar to Integer, except that any value added to an attribute is arithmetically added to the total, and any value deleted is arithmetically subtracted from the total.
Distinguished Name	The attribute is the distinguished name of the object up to 256 Unicode characters. This is not case sensitive.
EMail Address	Used to match attributes whose values are e-mail addresses and whose lengths and corresponding characters are identical; however, it ignores case (upper and lower). Only the EMail Address attribute uses this syntax.
Facsimile Telephone Number	Facsimile Telephone Number values are matched based on the **telephone number** field. The rules for matching fax telephone numbers are identical to those for the Case Exact syntax except that all space and hyphen (-) characters are skipped during the comparison. Only the Facsimile Telephone Number attribute uses this syntax.
Hold	This syntax is an accounting quantity, which is an amount tentatively held against a subject's credit limit, pending completion of a transaction. In the wire format, the **Subject** field is the distinguished name of the object. The Identity Vault treats the Hold amount similarly to the Counter syntax, with new values added to or subtracted from the base total. If the evaluated Hold amount goes to 0 (zero), the Hold record is deleted.
Integer	The attribute is an integer. Attributes using this syntax can set size limits.
Interval	The Interval value is the number of seconds in a time interval.
Net Address	Stores the network address as a binary string. The string is the literal value of the address. It lists the type of communication protocol used.
Numeric String	Two numeric strings match for equality when they are of the same length and their corresponding characters are identical. It matches the digits 0-9 and spaces if they are contained in the numeric string.
Object ACL	An Object ACL value can protect either an object or an attribute. The protected object is always the one that contains the ACL attribute. If an ACL entry is to apply to the object as a whole, the protected attribute name should be left empty (NULL). If a specific attribute is to be protected, it should be named in the ACL entry.
Octet List	A presented octet list matches a stored list if the presented list is a subset of the stored list. Octet strings are so designated because they are not interpreted by the Directory. They are simply a series of bits with no Unicode implications. The length is the number of bits divided by 8 and rounded to the nearest integer. Thus, each octet represents eight bits of data. The number of data bits is always evenly divisible by 8.
Octet String	For two octet strings to match, they must be the same length and the corresponding bit sequence (octets) must be identical. When comparing two strings, the first pair of octets that do not match are used to determine the order of the strings. Octet strings are not Unicode strings.

Syntax	Description
Path	The string represented by the path field is compared for equality by using the same rules that Case Exact String uses. That is, two paths match for equality when their lengths and corresponding characters, including case, are identical.
Postal Address	An attribute value for Postal Address is typically composed of selected attributes from the MHS Unformatted Postal O/R Address version 1 according to Recommendation F.401. The value is limited to 6 lines of 30 characters each, including a Postal Country Name. Normally the information contained in such an address could include a name, street address, city, state or province, postal code, and possibly a postal office box number depending on the specific requirements of the named object.
Printable String	The following characters are in the printable string character set. A...Z a...z 0...9 Space Character ‘ Apostrophe (Left Parenthesis) Right Parenthesis + Plus Sign Modeler, Comma - Hyphen . Period / Slash : Colon = Equal Sign ? Question Mark Two printable strings match for equality when they are the same length and their corresponding characters are identical. Case (upper or lower) is significant when comparing printable strings. For example, as printable strings, “Jones” and “JONES” do not match.
Replica Pointer	Each value of the replica pointer syntax is composed of five parts: ◆ The complete name of the server that stores the replica. ◆ A value describing the capabilities of this copy of the partition: master, secondary, read-only, or subordinate reference. ◆ A value indicating the current state of the replica (new, dying, locked, changing state, splitting, joining, or moving). ◆ A number representing the replica. All replicas for a partition have a different number assigned when the replica is created. ◆ A referral that contains a count of the addresses and one or more network addresses that hints at the node where the server probably resides. Because servers are accessible over different protocols, the server might have an address for each supported protocol.
Stream	Streams are files of information. The data stored in a stream file has no syntax enforcement of any kind. It is purely arbitrary data, defined by the application that created and uses it. The attribute is single valued.
Telephone Number	The length of telephone number strings must be between 1 and 32. Two telephone numbers string match for equality when they are of the same length and their corresponding characters are identical. All spaces and hyphen (-) characters are skipped during the comparison.
Time	A time value consists of a whole number of seconds, where zero equals 12:00 midnight, January 1, 1970, UTC.

Syntax	Description
Timestamp	A Timestamp value contains three components: ◆ The **wholeSeconds** field consists of the whole number of seconds, where zero equals 12:00 midnight, January 1, 1970, UTC. ◆ The **replicaNum** field identifies the server that created the Timestamp. A replica number is assigned whenever a replica is created on a server. ◆ The **eventID** field is an integer that orders events occurring within the same whole-second interval. The event number restarts at one for each new second.
Typed Name	The syntax names an Identity Vault object and attaches two numeric values to it: ◆ The level of the attribute indicates the priority. ◆ The interval indicates the frequency of references. The objectName or Distinguished Name identifies the Identity Vault object referred to by the Typed Name.
Unknown	Unknown syntax is used to stop the loss of data, if the Identity Vault database becomes corrupted. When an object becomes Unknown, there is information stored in this attribute that can allow the object to be recovered. This syntax is used by the Identity Vault.

NOTE: The information in this table comes from the LogicSource for eDirectory. LogicSource is a subscription-based service OpenText provides to its customers. For more information about LogicSource, see Technical Subscriptions (http://support.novell.com/subscriptions/articles/novell_logicsource.html).

Show Inherited Associations

The **Show Inherited Associations** check box determines whether the Associations list displays all classes associated with an attribute. When this check box is selected (the default), the Associations list displays both assigned and inherited classes. When this check box is deselected, the Associations list displays only assigned classes.

The schema allows for inheritance of other attributes from superclasses. If you select this item, all attributes that are associated with a class, whether assigned or inherited, are listed. If you don't select this item, only the assigned attributes are listed.

Used by Classes lists all classes that use the selected attribute. If you select **Show inherited associations**, the list includes classes that inherit the attribute.

Associations List

The Associations list displays the classes associated with the selected attribute. The Associations list toolbar lets you make changes to the classes associated with the attribute.

Class Field	Description
Add as Naming	Associates the selected attribute as a naming attribute to a class.
Add as Mandatory	Associates the selected attribute as a mandatory attribute to a class.
Add Optional	Associates the selected attribute as an optional attribute to a class.
Delete	Deletes the selected classes from the association list.

Creating Classes and Attributes

Designer allows you to create Identity Vault classes and attributes to fit the needs of your environment. You can test and use the new schema with the OpenText Identity Manager drivers in Designer before implementing it in the production environment.

- “Creating Identity Vault Classes” on page 148
- “Creating Identity Vault Attributes” on page 149

Creating Identity Vault Classes

- “Adding a Class” on page 148
- “Adding a Note” on page 149

Adding a Class

1. In the Modeler, right-click the Identity Vault, then select **Manage Vault Schema**.

   The **Classes** tab lists all classes that are defined in the schema and stored in Designer. For more information about the **Classes** tab, see “The Classes Tab” on page 138.
2. Select the **Add a Class** icon.
3. In the Create Class Name dialog box, specify the class name (for example, EmpInfo) and ASN1 ID (if applicable), then click **Next**.

   For more information about ASN1 IDs, see “ASN1” on page 139.
4. In the Class Flags dialog box, select the class type, then click **Next**.

   For information about the class type options, see Table 5-2 on page 140.
5. In the Class Inheritance dialog box, select the classes from which the new class inherits, then click **Next**.

   Select one or more classes in the **Available classes** list and use the right-arrow icon to move them to the **Inherited classes** list. Use the left-arrow icon to remove classes from the **Inherited classes** list using the left-arrow icons.
6. In the Mandatory Attributes dialog box, select the mandatory attributes, then click **Next**.

The inherited attributes displayed in the **Inherited mandatory attributes** pane depend upon the classes from which the new class inherits.

7 In the Optional Attributes dialog box, select optional attributes, then click **Next**.

The **Inherited optional attributes** pane lists default optional inheritances.

8 In the Naming Attributes dialog box, select the naming attributes, then click **Next**.

The Identity Vault schema allows for inheritance from other classes. A class that another class inherits from is called a superclass. A class can inherit attributes from one or more superclasses.

Every class inherits from the superclass Top. No class exists above Top. For example, Group inherits directly from Top, but User inherits from Organizational Person. Organizational Person inherits from Person. Person inherits from ndsLoginProperties, and ndsLoginProperties inherits from Top.

9 In the Containment Classes dialog box, select the containment classes for the new class, then click **Next**.

This specifies the types of container classes that can contain the new class. For example, if you select the class Group, the Manage Schema tool lists Domain, Organization, and Organizational Unit classes as containment classes for the Group class

10 In the New Class Summary, review the new class information, then click **Finish**.

The new class appears in the Classes pane.

11 Click **OK** to save changes and close the Manage Schema tool.

Adding a Note

Designer allows you to add notes about any class you create. The information is stored as `desc` in the `.ldif` file and as a note in the `.sch` file.

1 Select the class you want to add a note to, then click the **Schema Notes** icon .

2 Type the note in the window, then click **OK**.

Creating Identity Vault Attributes

To create a new Identity Vault attribute:

1 In the Modeler, right-click the Identity Vault, then select **Manage Vault Schema**.

2 Select the **Attributes** tab.

The Attributes list displays all attributes that are defined in the schema and stored in Designer. You can view all attributes at once, or view the attributes associated with a specific class by selecting a class from the drop-down list.

For more information about the components of the **Attributes** tab, see “The Attributes Tab” on page 142.

3 Select the **Add an Attribute** icon .

4 In the Create Attribute Name dialog box, specify the attribute name (for example, EmpID) and an ASN1 ID, if applicable, then click **Next**.

For more information about the ASN1 ID, see “ASN1” on page 144.

5 In the Attribute Syntax dialog box, select the proper attribute syntax, then click **Next**.

An attribute syntax defines a standard data type that an attribute uses to store its values in the Identity Vault. Each attribute must have a syntax. See "Syntax" on page 144 for more information.

6 In the Attribute Flags dialog box, select the flags for the attribute, then click **Next**.

Attribute flags constrain the information that is stored in the attribute, and the list of acceptable operations that the Identity Vault, and Identity Vault clients, can perform on the attribute. For more information about attribute flags, see "Flags" on page 143.

7 In the New Attribute Summary dialog box, review the new attribute information, then click **Finish**.

The new attribute appears in the Attributes list.

8 Click **OK** to save changes and close the Manage Schema tool.

Modifying the Schema

Designer allows you to modify the Identity Vault schema. The following sections describe fields and definitions used in the Manage Schema tool for classes and attributes.

- "Deleting Schema Definitions" on page 150
- "Modifying Classes or Attributes" on page 150
- "Renaming Schema Definitions" on page 150

Deleting Schema Definitions

You can delete an extended schema definition. You cannot delete base schema elements. If you select a base schema class or attribute, the **Delete** icon is disabled.

1 In the Modeler, right-click an Identity Vault, then select **Manage Schema**.

2 Select the class or attribute that you want to delete, then click the **Delete** icon .

Modifying Classes or Attributes

1 In the Modeler, right-click an Identity Vault, then select **Manage Vault Schema**.

2 Select the class or attribute that you want to modify.

3 Modify the class or attribute as desired.

If you select a base schema class or attribute, a warning message appears.

It is best to modify only the extended schema and not the base schema. Modifying the base schema can cause data corruption and synchronization errors.

Renaming Schema Definitions

You can rename extended schema definitions. You cannot rename any base schema classes or attributes. If you select a base schema item, the **Rename** icon is dimmed, indicating it is unavailable.

- "Renaming a Class" on page 151
- "Renaming an Attribute" on page 151

Renaming a Class

1. In the Modeler, right-click an Identity Vault, then select **Manage Vault Schema**.
2. In the **Class** page, select a class that you want to rename, then click the **Rename Class** icon .
3. In the Rename Class dialog box, specify the new class name, then click **OK**.

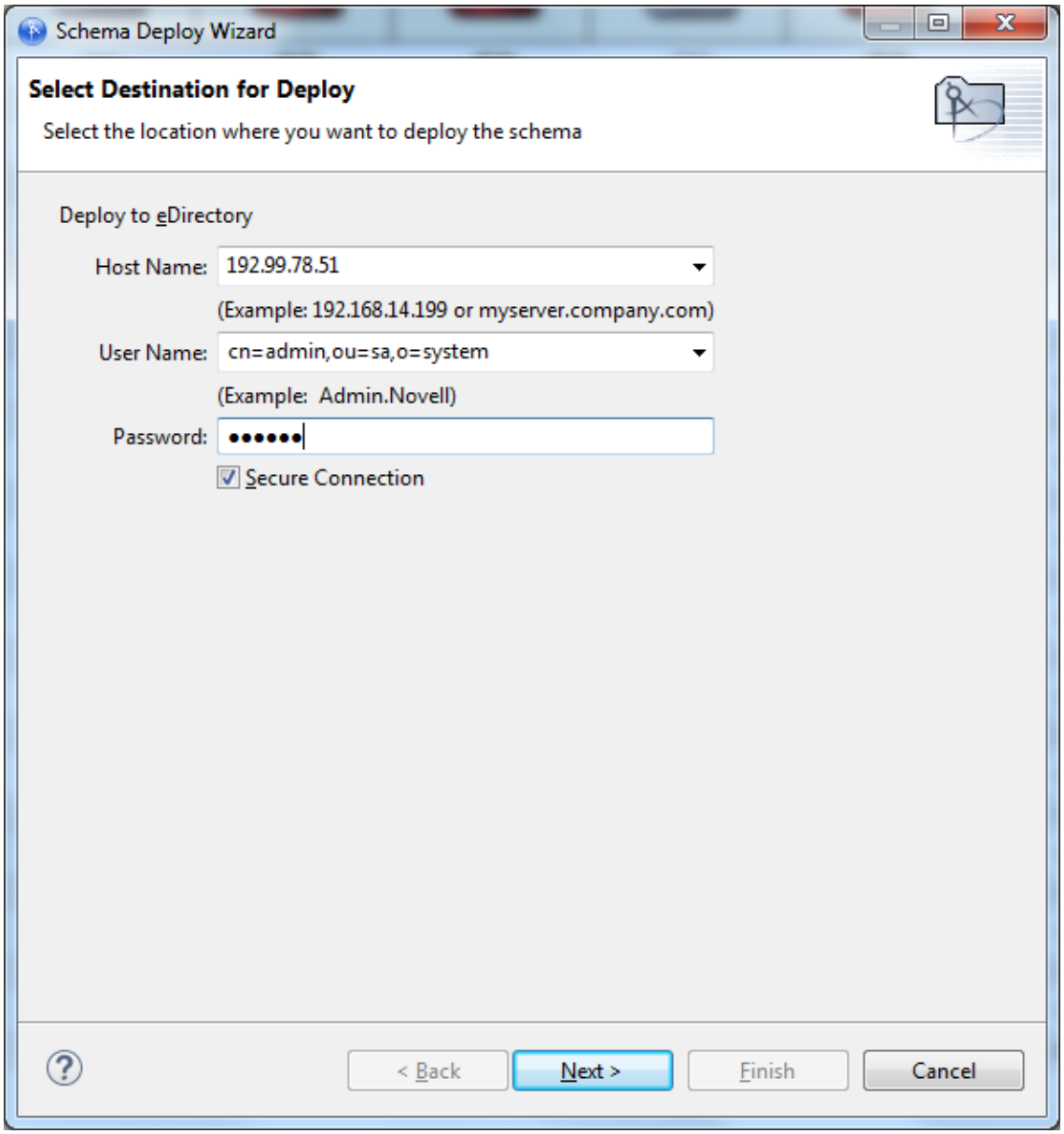
Renaming an Attribute

1. In the Modeler, right-click the **Identity Vault** icon, then select **Manage Vault Schema**.
2. Select an attribute you want to rename in the **Attribute** tab, then click the **Rename an Attribute** icon .
3. In the Rename Attribute dialog box, specify the new attribute name, then click **OK**.

Deploying the Schema into the Identity Vault

After the OpenText Identity Manager driver is tested with the new schema, you can deploy the modified schema into the Identity Vault.

1. In the Modeler, select the Identity Vault.
2. Select **Live > Schema > Deploy**.
3. Specify the **Host Name**.



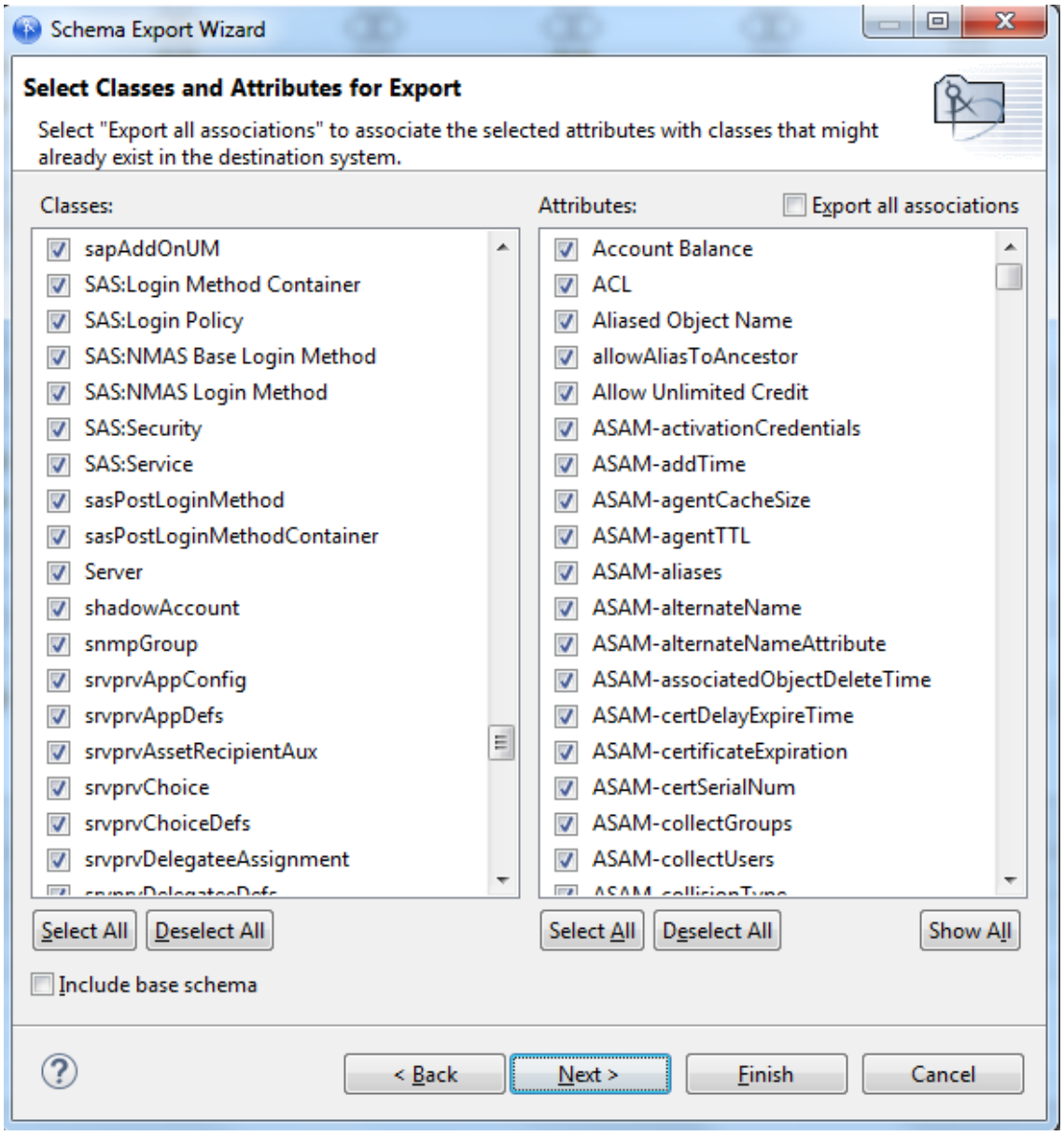
The host name can be the server's IP address or the DNS name of the server.

4 Specify the **User Name** in LDAP format, which must be a user with administrative rights to the schema.

5 Specify the user's password, then click **Next**.

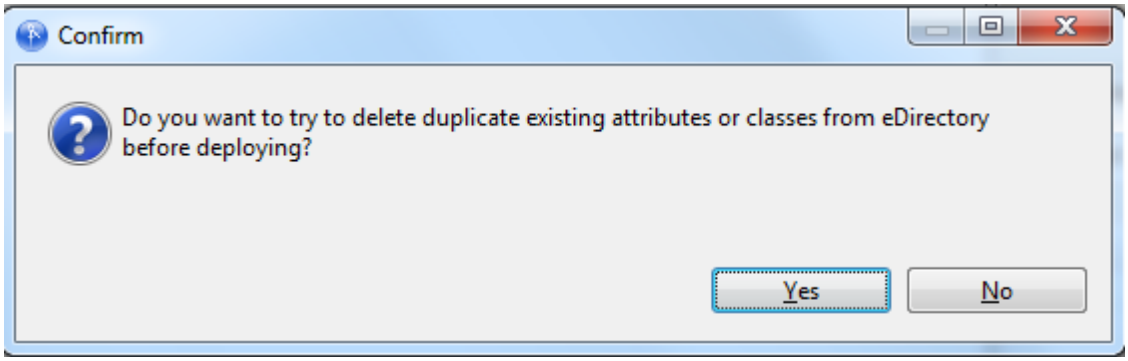
While connecting to the live Identity Vault, Designer prompts you to ensure that the certificate that you are accepting is valid. You can choose to accept the certificate temporarily for the session or accept it permanently. You can instruct Designer to remember this setting for future authentication to the Identity Vault.

6 Select the classes and attributes to deploy into the Identity Vault schema, then click **Next**.



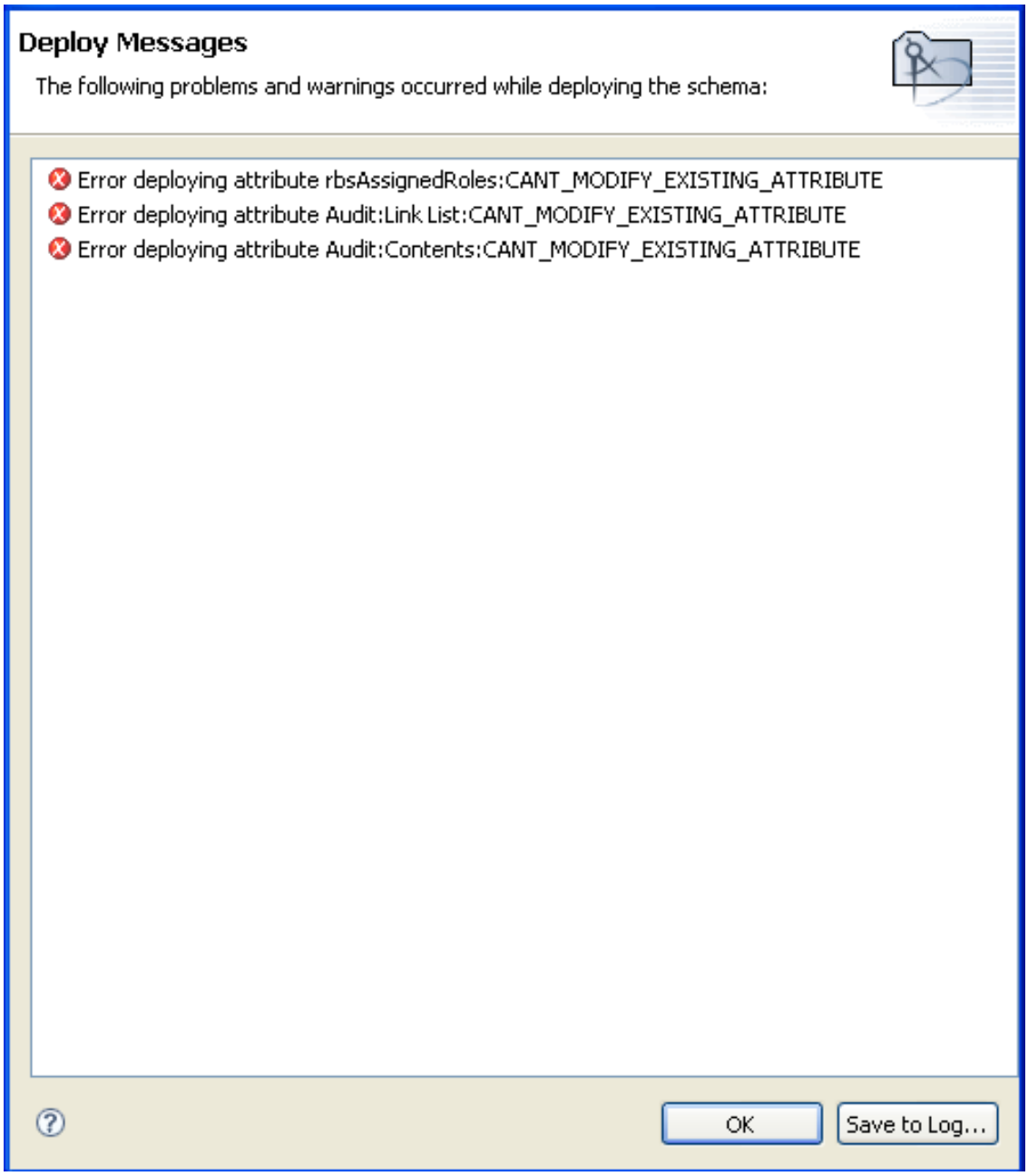
7 Review the summary of classes and attributes to be deployed, then click **Finish**.

If you have selected duplicate attributes or classes, a warning box appears



8 Select **Yes** or **No**, depending upon whether you want to resolve the duplicate classes or attributes.

9 Review errors or warnings, then click **OK**.



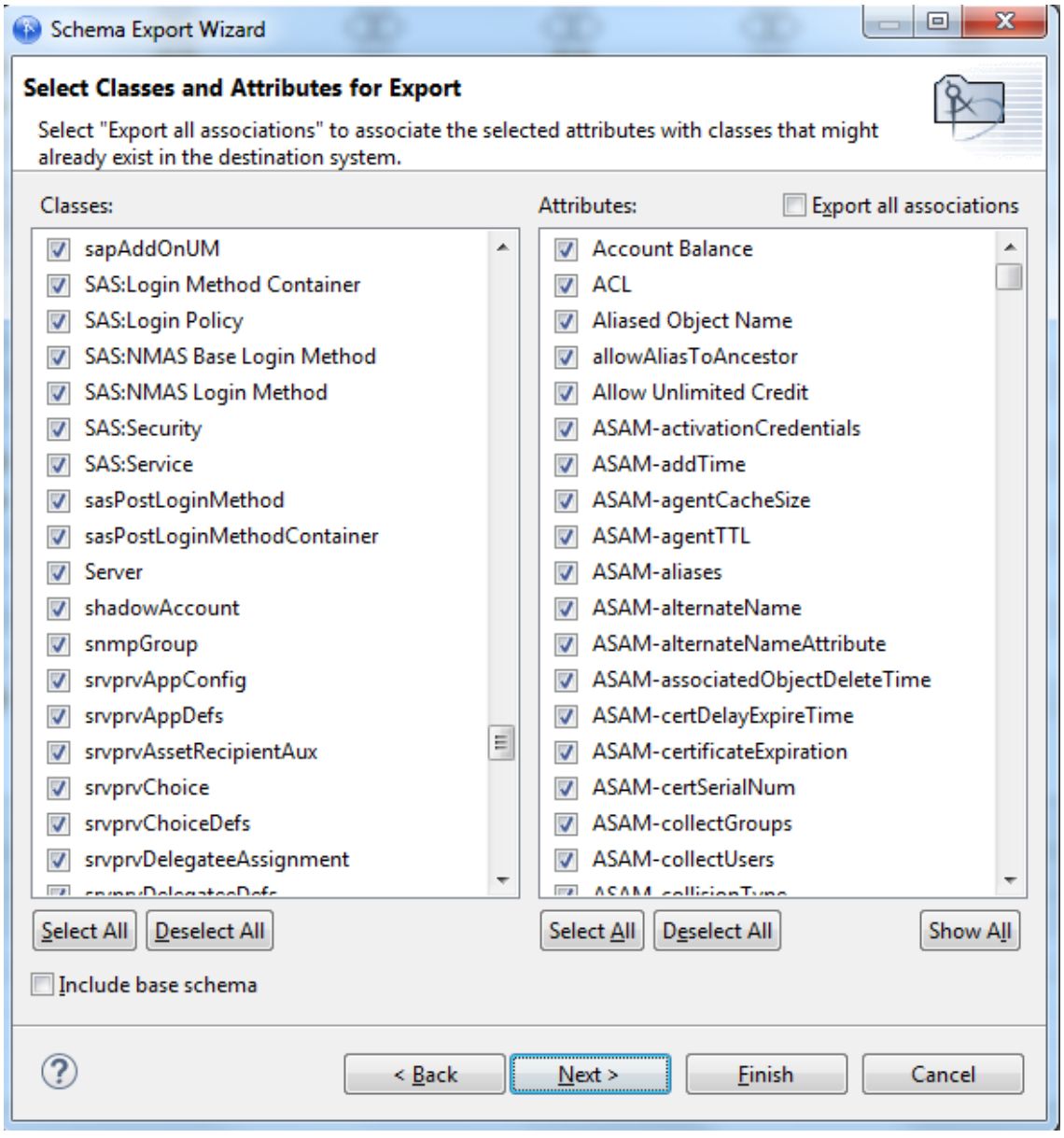
Exporting the Schema to a File

- "Exporting the Schema to a .sch File" on page 155
- "Exporting the Schema to an LDIF File" on page 156

Exporting the Schema to a .sch File

1. In the Modeler, right-click an Identity Vault, then select **Export to File > Schema**.
2. In the Schema Export Wizard, select **.sch format**.
3. Specify a filename and location where you want to save the schema file, then click **Next**.

   Designer appends the .sch extension when you export the file.
4. In the Select Classes and Attributes for Export page, select the classes and attributes to export to the .sch file.



Export all associations (above the **Attributes** pane) enables you to associate the selected attributes with the classes that might already exist in the Identity Vault. If you do not select this box, the new attributes that should be associated with the class are not associated.

For example, if the Employee Photo attribute is associated with the User class, and **Export all associations** is not selected, Employee Photo is not associated with the User class.

The classes and attributes that are in Designer are listed in the two columns. All classes and attributes are selected by default. To prevent a class or attribute from being deployed, deselect it. To add all classes and attributes, click **Select All**. To remove all classes and attributes, click **Deselect All**.

5 When you have finished selecting classes and attributes, click **Finish.**

Exporting the Schema to an LDIF File

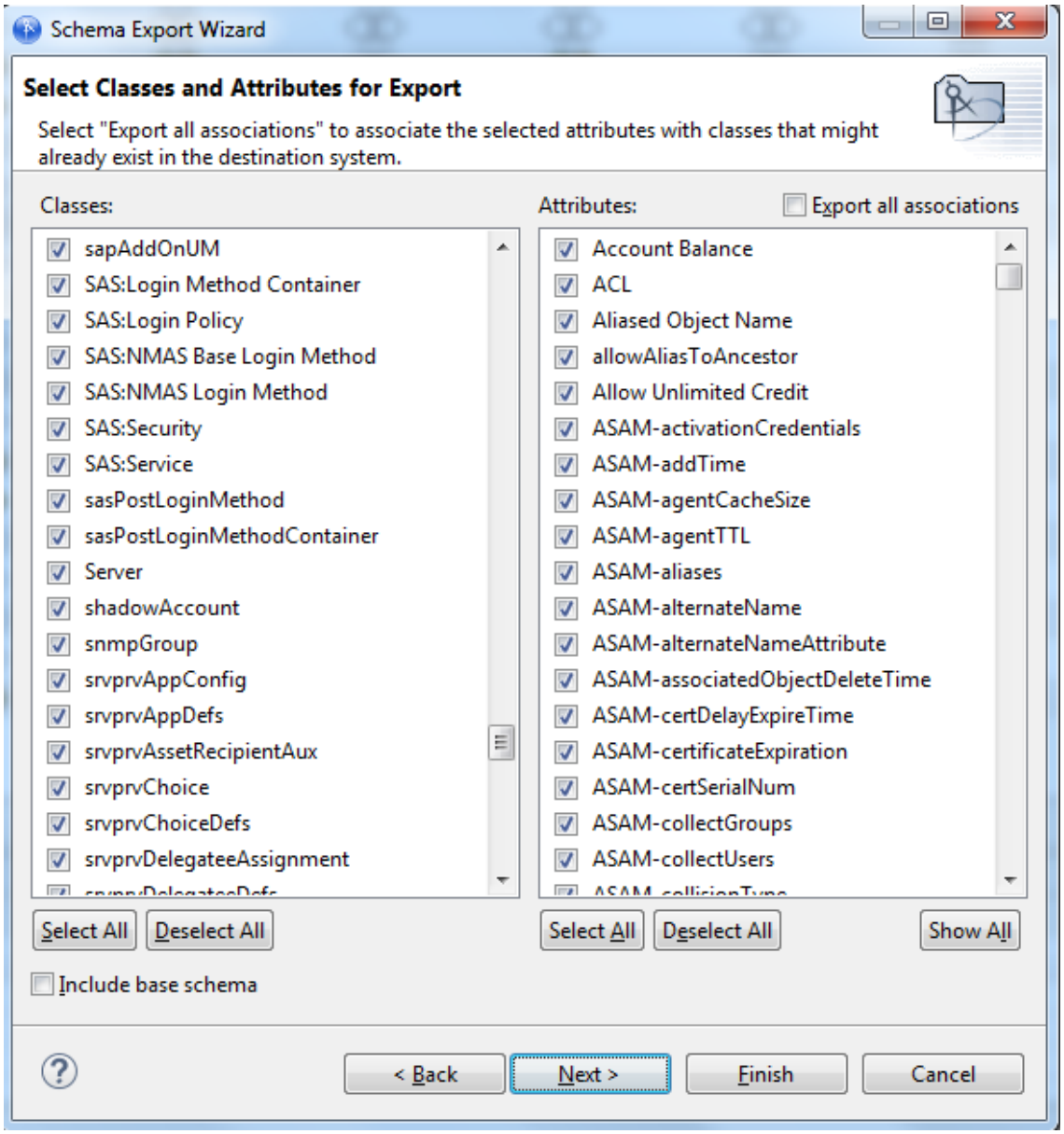
1 In the Modeler, right-click the Identity Vault, then select **Export to File > Schema**.

2 In the Schema Export Wizard, select **.ldif format**.

3 Specify a name and location where you want to save the schema file, then click **Next**.

Designer appends the `.ldif` extension when you export the file.

4 In the Select Classes and Attributes for Export page, select the classes and attributes to export to the `.ldif` file.



Export all associations enables you to associate the selected attributes with the classes that might already exist in the Identity Vault. If you do not select this box, the new attributes that should be associated with the class are not associated.

For example, if the Employee Photo attribute is associated with the User class, and **Export all associations** is not selected, Employee Photo is not associated with the User class.

5 When you have finished selecting classes and attributes, click **Finish.**

6 Click **OK** in the warning.

The class and attribute names in Designer are the Identity Vault (OpenText eDirectory) names. The names for the classes and attributes in the LDIF file are the LDAP names. The Identity Vault names differ from the LDAP names. Verify that the names listed in the LDAP file are correct for your environment before importing the file. For a list of Identity Vault class and attribute names mapped to LDAP class and attribute names, see "Mapping Identity Vault to an LDAP Schema" on page 167.

Importing the Schema

Designer allows you to import the schema from your production environment to do in-depth testing with the OpenText Identity Manager drivers.

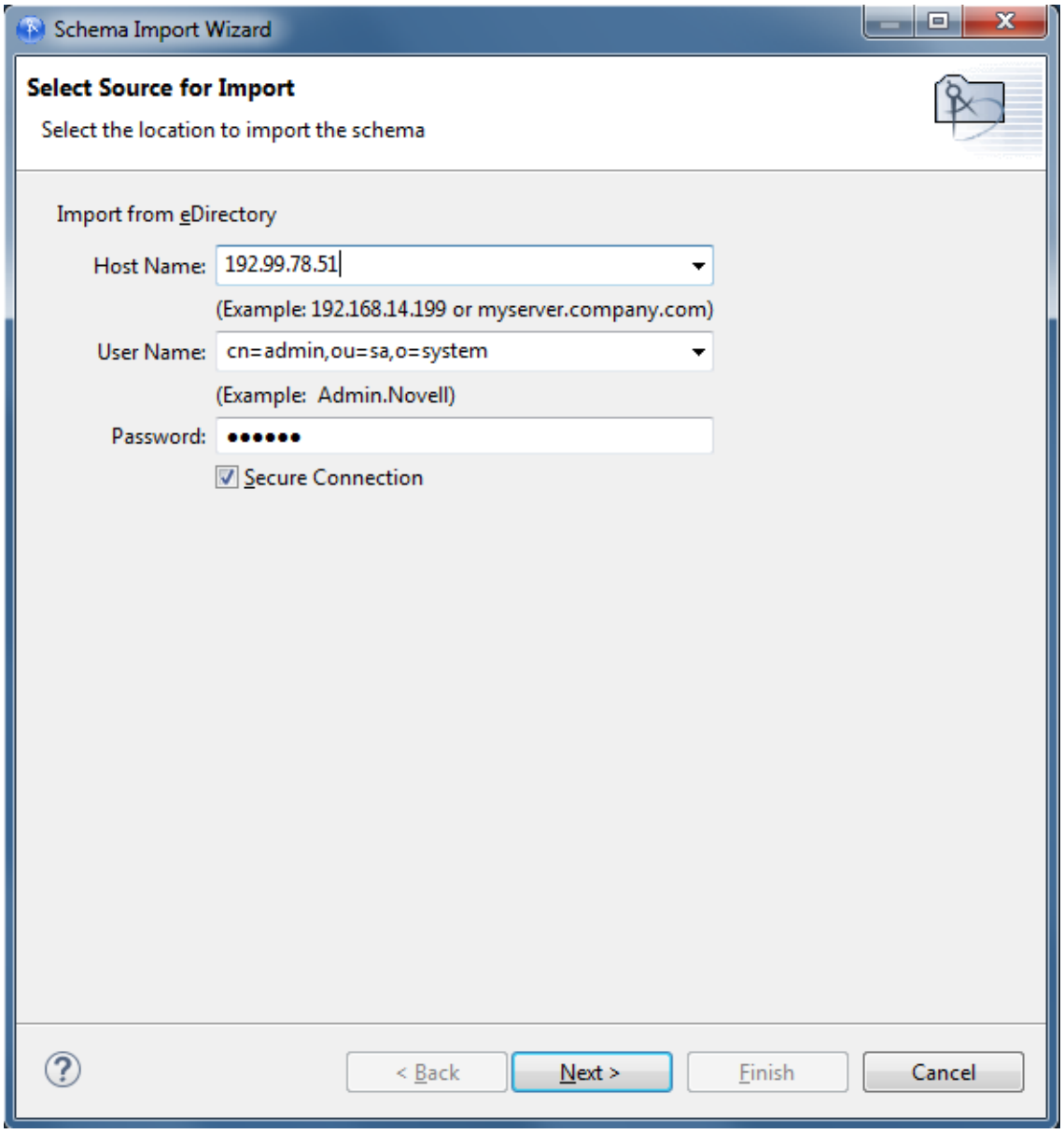
- “Importing the Schema from the Identity Vault” on page 158
- “Importing the Schema from a File” on page 162

Importing the Schema from the Identity Vault

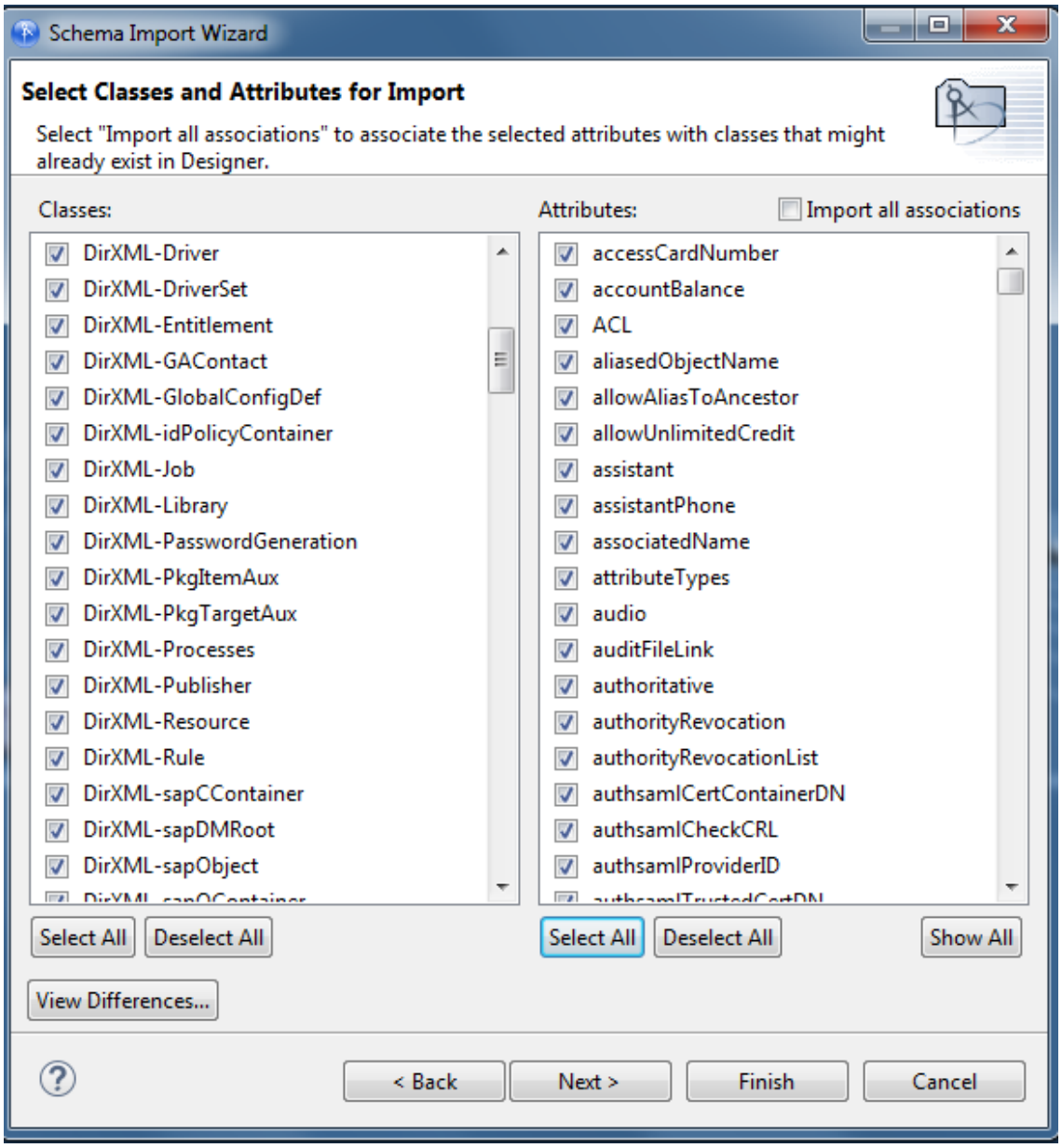
1. In Designer, select an Identity Vault, then select **Live > Schema > Import**.
2. In the Select Source for Import dialog box, specify the access information to access the server that has the schema to import, then click **Next**.

   Specify the appropriate host name (or IP address), username, and password to access the server.

   NOTE: The specified user must have administrative rights to the schema.



3 In the Select Classes and Attributes for Import page, select the classes and attributes to import into the project, then click **Next**.



Import All Associations: Enables you to associate the selected attributes with the classes that might already exist in Designer. If you do not select this box, the new attributes that should be associated with the class are not associated.

For example, if the attribute of Employee Photo is associated with the User class, and you do not select **Import all associations**, Employee Photo is not associated with the User class.

View Differences: Enables you to view the differences in the schema between the Identity Vault and Designer.

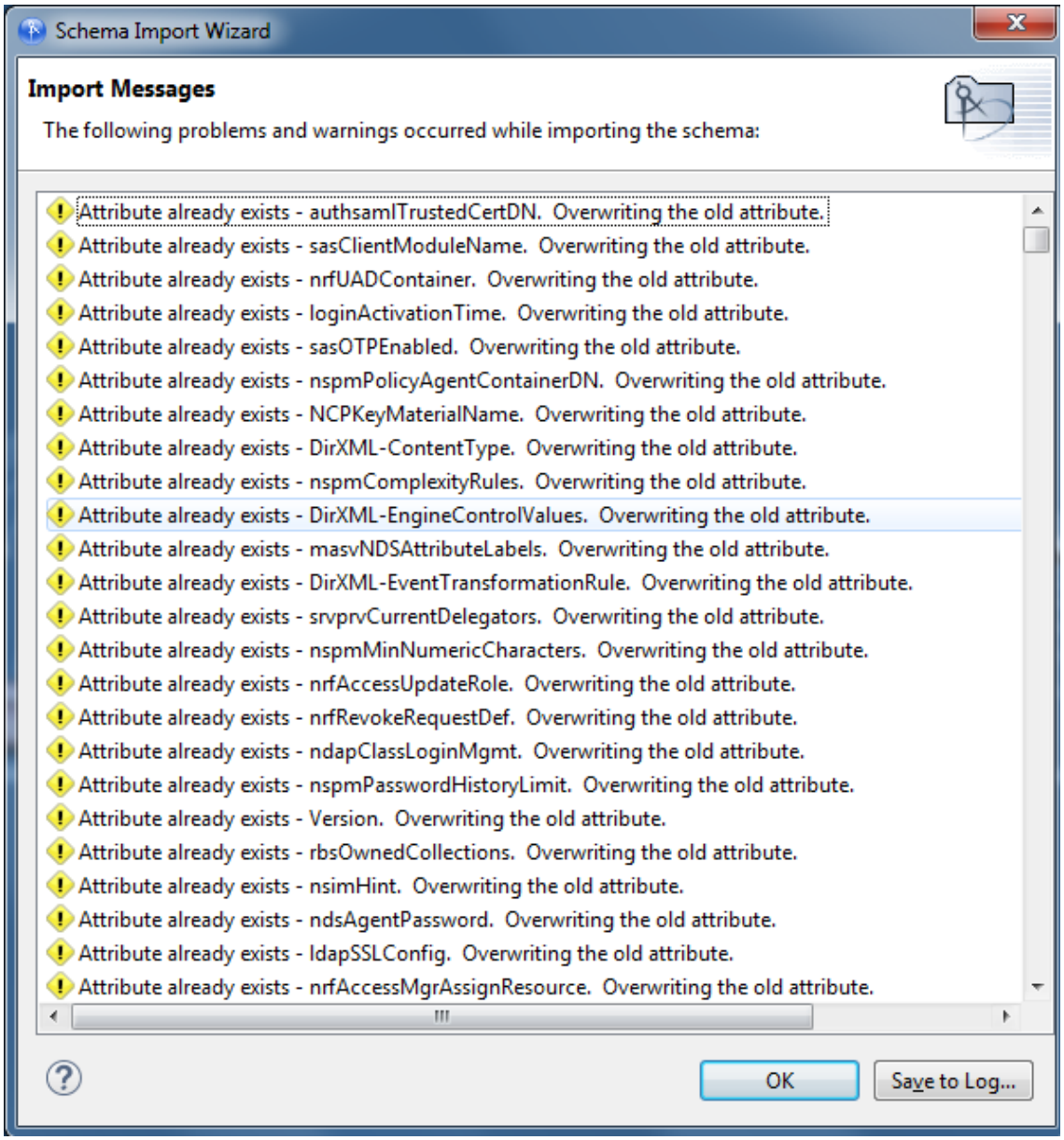
When you click **View Differences**, Designer opens the Schema Differences page, where you can select those differences between the live Identity Vault and the Identity Vault in your project.

You can select schema differences individually, or click **Select All** to import all the schema differences.

4 Click **OK** to move the selected class and attribute import selections into the Select Classes and Attributes for Import page.

5 Click **Next** to bring up the Import Summary page, where you can review classes and attributes to import into the project. Then click **Finish**.

If errors occur during the import process, the Import Messages page lists them.



6 On the Import Messages page of the Schema Import Wizard, click **OK**.

or

If you want to save the differences to a log file, click **Save to Log**. This brings up the Save As dialog box, where you can choose a filename and directory to store the file in.

7 Click **Save**, then click **OK**.

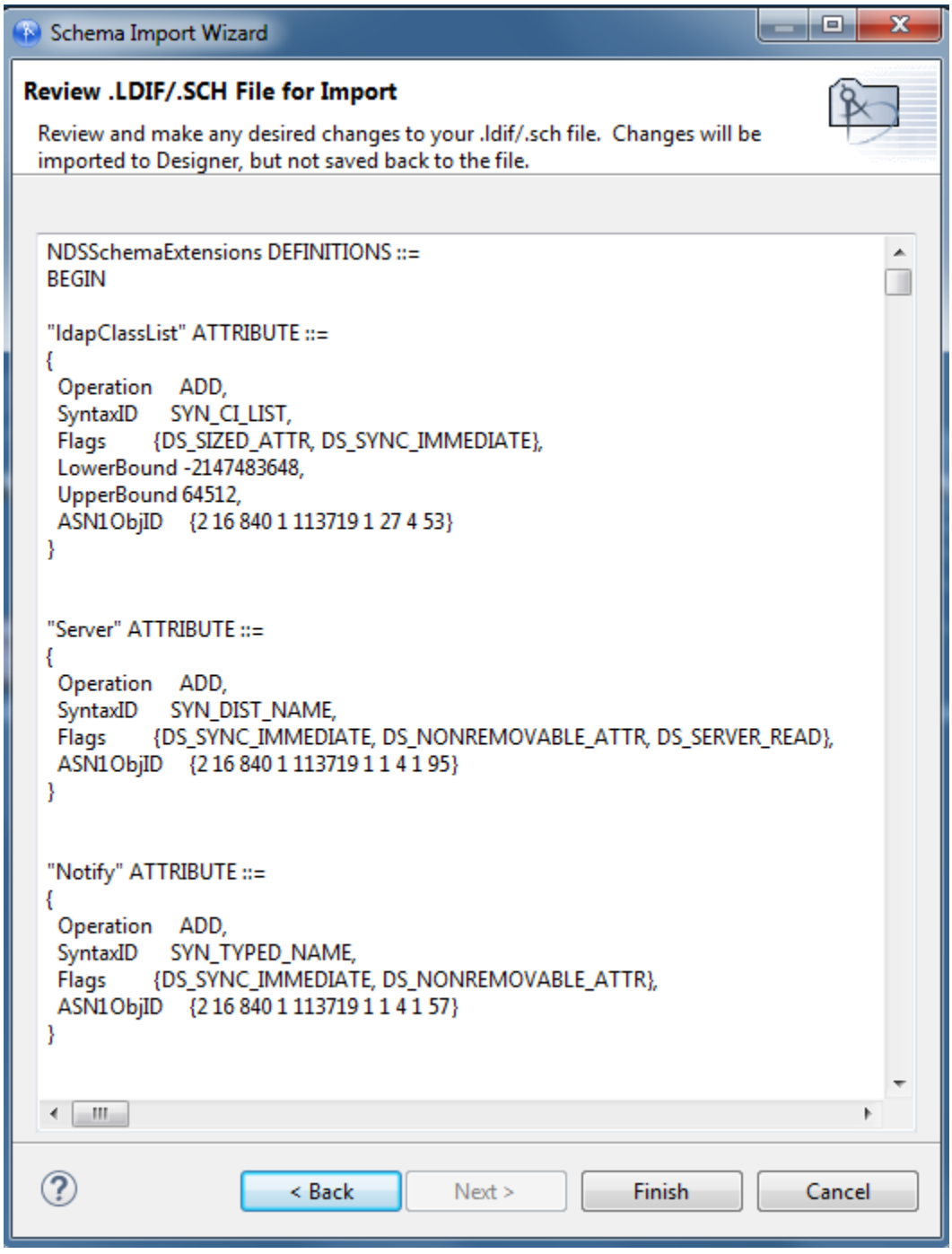
Importing the Schema from a File

When you created an Identity Vault in the Modeler, Designer created a base schema in your project. If a .sch file or .ldif file has been saved, you can quickly add classes and attributes for your drivers by importing classes and attributes from the saved file.

- "Importing the Schema from a .sch File" on page 162
- "Importing the Schema from an LDIF File" on page 164

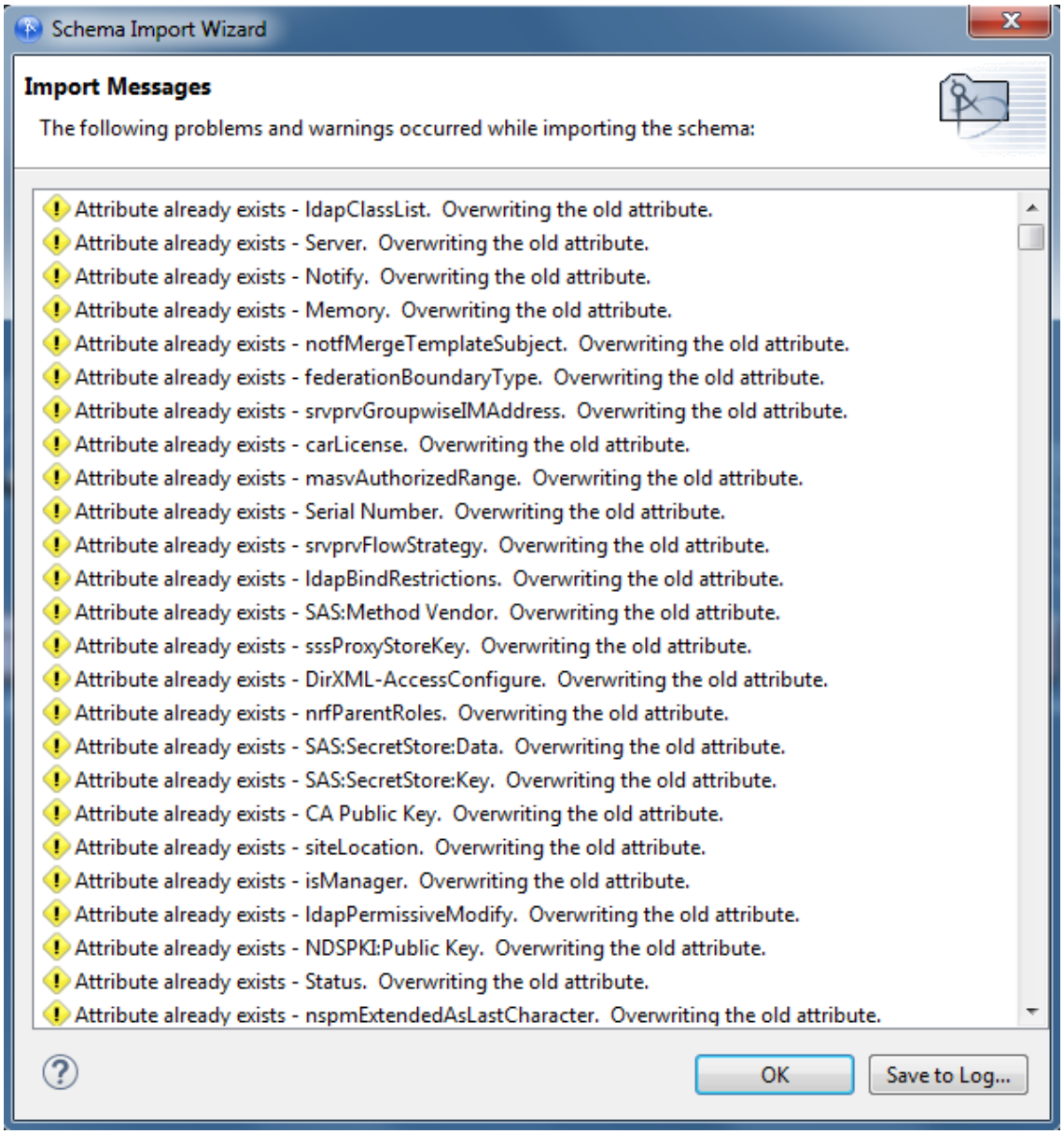
Importing the Schema from a .sch File

1. In the Modeler, right-click the Identity Vault that will use the imported .sch file.
2. Select **Import Schema from File**.
3. Select **.sch format.**
4. Browse to and select the **.sch** file that you want to use, then click **Open**.
5. Click **Next**, then review the .sch file.



6 Make changes if necessary, then click **Finish**.

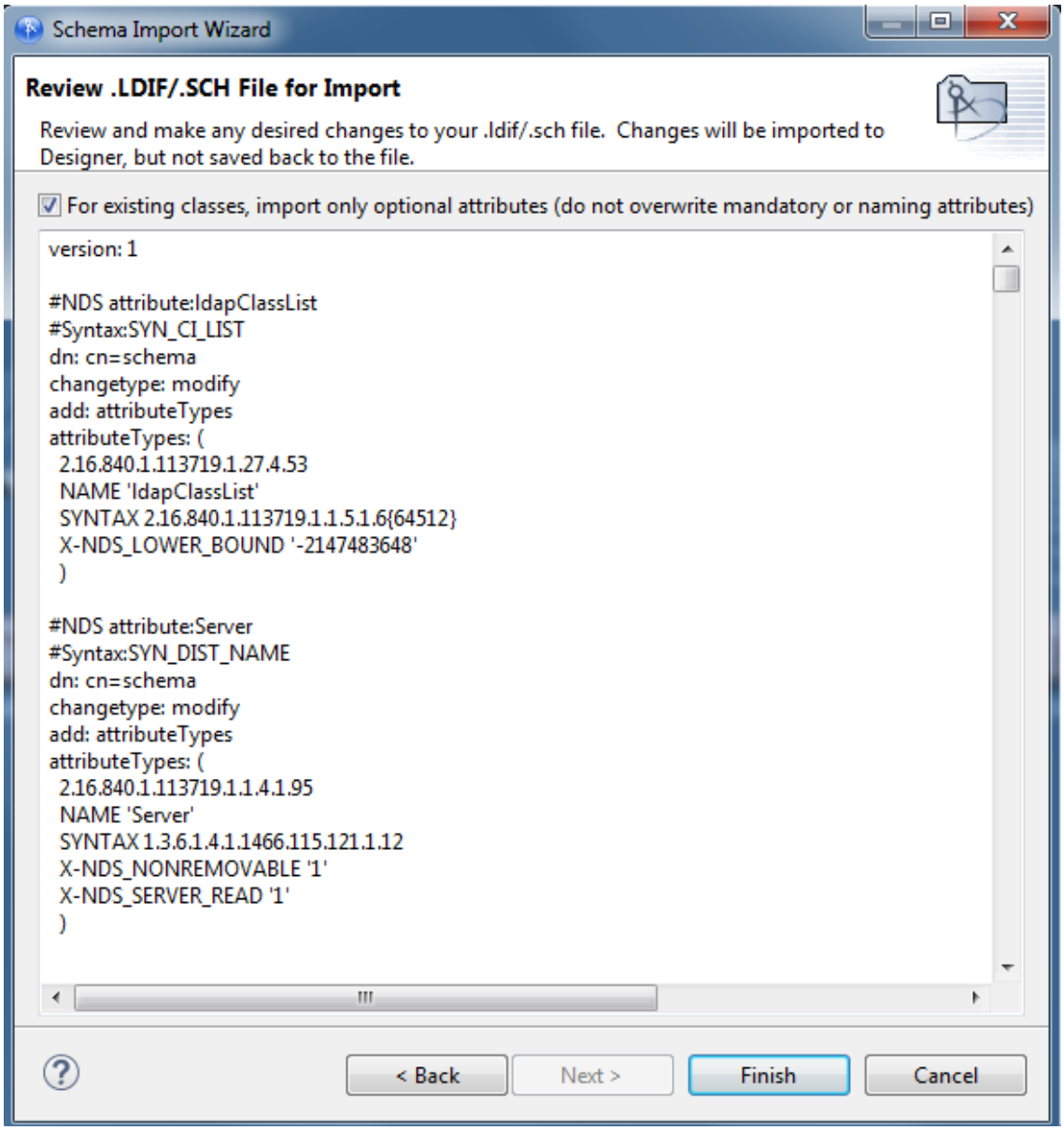
7 Click **OK**.



If errors occur, a deployment summary screen lists them.

Importing the Schema from an LDIF File

1. In the Modeler, right-click the Identity Vault that will use the imported `.ldif` file.
2. Select **Import Schema from File**.
3. Select **.ldif format**.
4. Specify, or browse to and select, the `.ldif` file that you want to use, then click **Open**.
5. Click **Next**, then review the `.ldif` file.

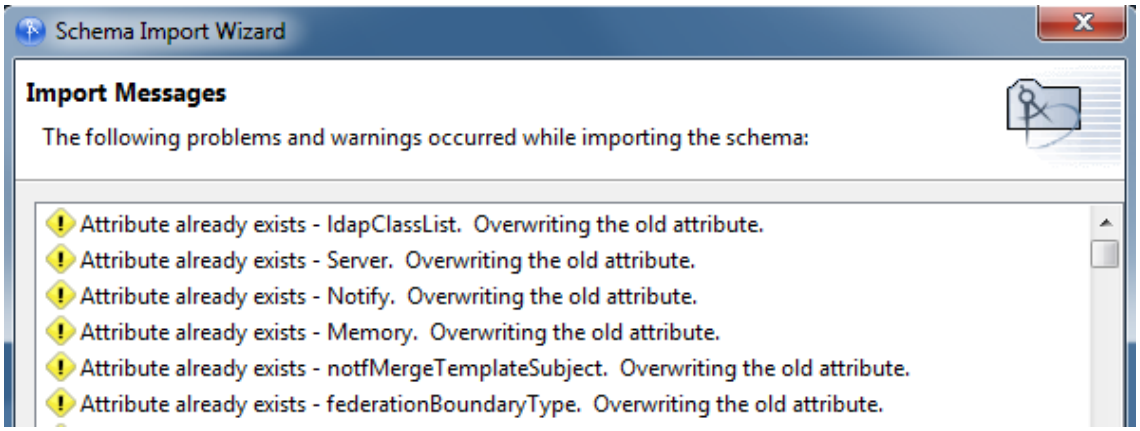


6 Make changes if necessary, then click **Finish**.

7 If you receive a Warning, read the message and click **OK**.

The class and attribute names in Designer are the Identity Vault (OpenText eDirectory) names. The names for the classes and attributes in the LDIF file are the LDAP names. The Identity Vault names differ from the LDAP names. Verify that the names listed in the LDAP file are correct for your environment before importing the file. For a list of Identity Vault class and attribute names mapped to LDAP class and attribute names, see "Mapping Identity Vault to an LDAP Schema" on page 167.

8 Click **OK**.



If errors occur, a deployment summary dialog box lists them.

Managing a Copy of an Application Schema

The OpenText Identity Manager engine currently uses the application schema for the following:

- DirXML Script uses the dn-format/dn-delims to figure out how to parse or convert DNs coming from and going to the application.
- To set the multi-valued flag on attributes that are used during the attribute merge process that happens as part of a match, resync, or migrate.

- "Editing an Application's Schema" on page 166
- "Refreshing the Application Schema" on page 167

Editing an Application's Schema

Designer enables you to manage a copy of the managed system's schema. You can make changes to a copy of the application schema so that you can test the OpenText Identity Manager drivers in Designer. The schema changes cannot be deployed into the live application schema.

1 Right-click the driver connection in Designer, then select **Manage Application Schema**.

2 Add, rename, or delete the application's classes or attributes, then click **OK**.

DN Format: Specifies the separator character used when specifying distinguished names. For example, admin.utah.novell.com.

Classes: Lists all of the classes stored in Designer from the application's schema.

Add a class: Adds a new class.

Rename class: Renames the selected class.

Delete class: Deletes the selected class.

Refresh application schema: Provides a new copy of the application's schema.This option is useful if the application schema changes.

Help: Launches the Help documentation for the Manage Schema tool.

Flags Container: Specifies whether the class is a container.

ASN1: The unique ID of the class.

Attributes of This Class: Lists all of the attributes stored in Designer for the selected class from the application's schema.

Add an attribute: Adds a new attribute for the selected class.

Rename attribute: Renames the selected attribute.

Delete attribute: Deletes the selected attribute.

Flags: Specifies the details of the attribute. To edit the flags, select an attribute.

Type: Specifies the syntax of the attribute. To view the syntax, select an attribute. To change the syntax, select an option from the drop-down list.

Refreshing the Application Schema

If the application schema changes, you can get a new copy of the application's schema by refreshing the application schema.

NOTE: An application schema is not automatically imported by default. You can always perform a refresh application schema operation on a particular application after the project has been imported.

---

1. Right-click the driver connection, then select **Live** > **Refresh Application Schema**.
2. Click the browse icon.
3. Browse to and select the server where the driver is installed, then click **OK** twice.

Mapping Identity Vault to an LDAP Schema

When you access the Identity Vault through LDAP, the names of classes and attributes might be different than when it is accessed through the standard NCP-based APIs.

For more information about how that mapping is performed, see Schema (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/fbaddhch.html) and "Class and Attribute Mappings" (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/h0000007.html#a5bwxyz) in the *OpenText™ eDirectory CE 24.4 (v9.3) eDirectory Administration Guide (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/bookinfo.html)*.

Comparing the Schema

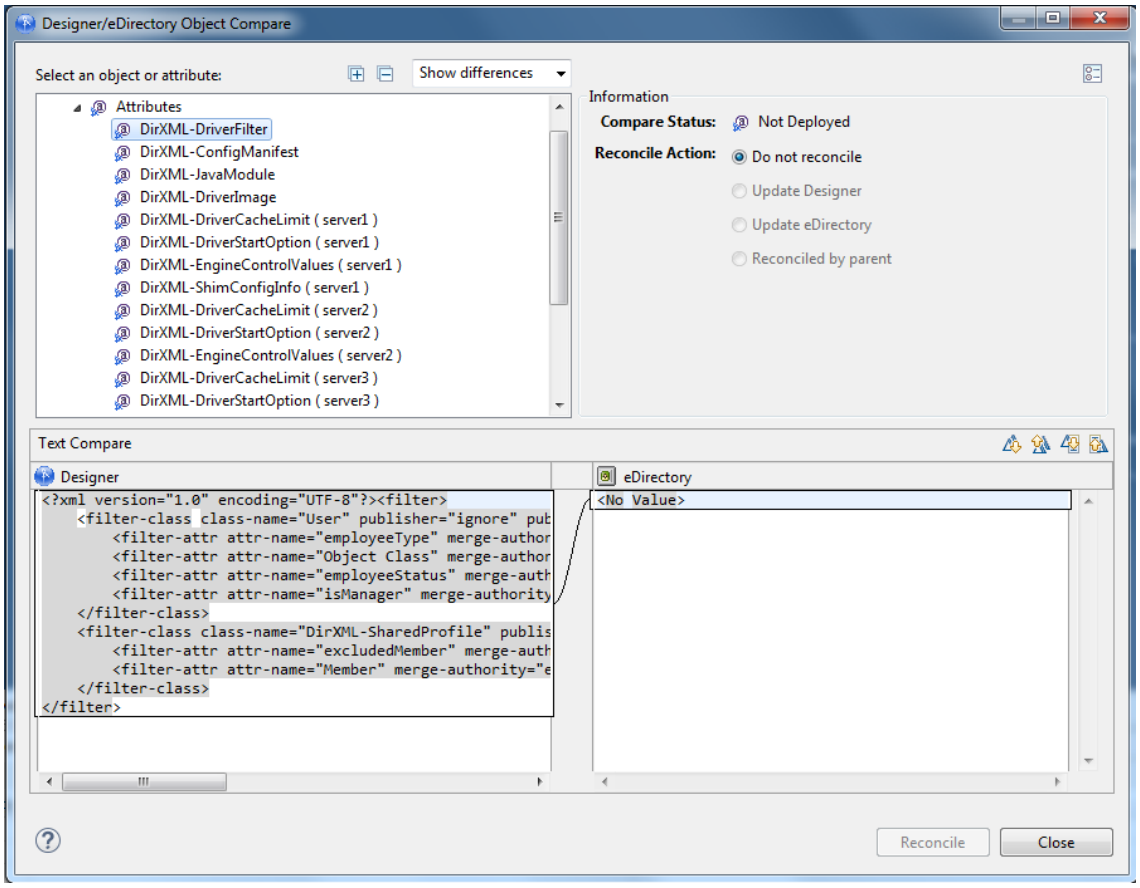
Designer allows you to compare schemas from your production environment to do in-depth testing with the OpenText Identity Manager drivers. Designer provides conflict resolution on individual classes and attributes and allows you to view the differences between existing and new values when importing and deploying the schema. For example, before deploying a schema to an Identity Vault, you can run Compare.

Compare shows whether the classes and attributes are equal (no action is necessary) or unequal. If they are unequal, you can choose not to reconcile them, choose to update them in Designer, or choose to update them in OpenText eDirectory.

You can run the Compare feature at any time. If you choose to reconcile the differences between schema in Designer and OpenText eDirectory while in Compare, you won't need to run Import or Deploy.

The following procedure assumes that you want to determine if you have deployed all the changes you made in the Designer schema to the Identity Vault schema.

1 Right-click the driver object in either the Modeler view or in the Outline view. Select **Live > Schema > Compare** to bring up the Designer/OpenText eDirectory Schema Compare window.



2 In the **Select a class or attribute** portion of the window, you see the listed classes and attributes. Select an individual class or an attribute to see the actual differences displayed in the **Text Compare** portion of the window.

The plus icon at the right side of the **Select a class or attribute** allows you to expand all elements in the parent object, and the minus icon collapses all of the elements. The ? icon displays the Summary/Compare dialog box help.

3 By default, the Compare window only displays values that are different between OpenText eDirectory and Designer. To view all the classes and attributes, select **Show all** from the pull-down menu. Your choices are **Show differences**, **Show deletes**, and **Show all**.

4 Check to see the status of the values that are shown. Values that are equal are shown as **Equal** on the **Compare Status** line in the Information portion of the Compare window.

The overlay image displayed in the **Compare Status** entry identifies objects or attributes that need reconciliation. The following table describes what you see in the **Compare Status** line and the overlays that you can see:

Compare Status	Description
Equal	The selected classes or attributes are same in OpenText eDirectory and Designer.
Unequal 	The value of the selected class or an attribute, or one or more classes or attributes, are different in OpenText eDirectory and Designer.
Not Deployed 	The selected class or an attribute is not deployed to OpenText eDirectory.
Not Imported	The selected class or an attribute does not exist in Designer.

5 Under the Information portion of the Compare window, select how you want to reconcile the differences between the Source and Destination.

If **Compare Status** shows **Unequal**, you have three choices:

- To do nothing, keep the default value of **Do Not Reconcile**.
- To update the driver in Designer so that it contains the same information as the driver in OpenText eDirectory, select **Update Designer**.
- To update the driver in OpenText eDirectory to reflect the changes you have just made to the driver in Designer, select **Update eDirectory**.

The green check box in the bottom corner of the icons shows all the child objects that are being reconciled with the parent object. If you select the parent object to perform the update, then all the child objects under the parent reflect that choice and you see the **Reconciled By Parent** icon selected. If you do not choose a parent object, you can reconcile each child object individually. You can also see a small Designer icon and an OpenText eDirectory icon, showing how objects are being reconciled.

6 Check to see the **Text Compare** values.

The **Text Compare** values displayed in the bottom portion of the Designer/OpenText eDirectory Schema Compare window shows the difference at the child object level. The Text Compare dialog box uses the Eclipse Compare editor to compare classes and attributes that contain XML data, such as policy data, driver filters, or configuration data. The differences in the code are highlighted in blue.

7 After you view the differences, click **Reconcile** to perform the reconciliation actions for each object in the tree, or click **Close** to close the Designer/OpenText eDirectory Object Schema Compare window.

6 Understanding Packages

OpenText Identity Manager drivers consist of multiple components like roles, workflows, policies, ECMAScripts, and style sheets. The configuration of each of these components makes each OpenText Identity Manager driver unique.

This complexity makes it challenging to add new content to drivers, as when you need to create different components multiple times. In order to save time and help manage OpenText Identity Manager content, version 4.0 and later includes a concept called packages.

For information about migrating driver configuration files to packages, see the OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Linux or OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Windows.

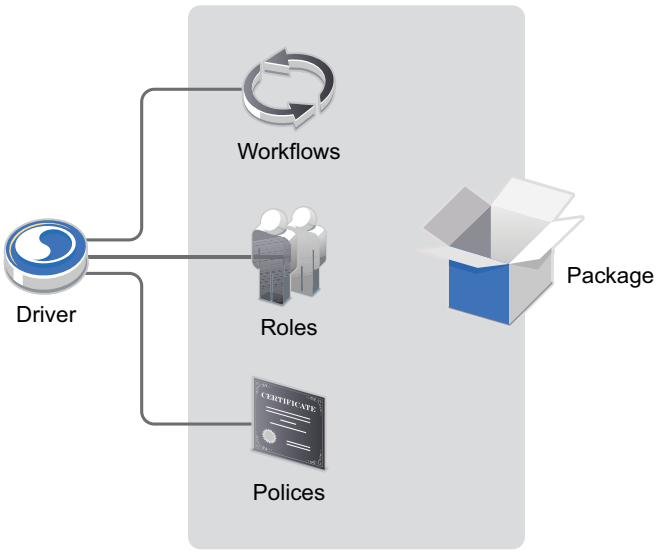
- "Using Packages" on page 171
- "Installing or Upgrading Packages" on page 175
- "Customizing Default Packages" on page 183
- "Managing Package Versions Using Git" on page 184
- "Removing or Downgrading Packages" on page 189

Using Packages

A package is a container for components of OpenText Identity Manager driver content, organized according to the functionality you want to provide to a driver. Packages can contain different types of content that you can move from one environment to another, allowing you to re-use content in multiple places and create and configure drivers more efficiently.

Designer allows you to export packages as .jar files. This enables you to easily share packages with other users and import packages into different instances of Designer.

Figure 6-1 *OpenText Identity Manager Package*



Designer allows you manage and develop packages. Packages are the delivery mechanism for OpenText Identity Manager content. When you create a package, you are creating the framework for delivering the content.

Packages are stored in the package catalog, which is only visible in Designer. The package catalog is created when you create or import a project and add an Identity Vault. If you have an existing project, the package catalog is created when you open the project after it is converted.

Developers can create packages to deliver custom content. For more information about developing packages, see Chapter 7, “Developing Packages,” on page 193.

Packages are only supported with version 4.0 or later. If you create a driver using a driver configuration file for an earlier version of OpenText Identity Manager, we recommend you migrate your existing driver to use packages.

For more information about how packages work, see the following sections:

- “Advantages of Packages” on page 172
- “Understanding Package Dependencies” on page 173
- “Package Content” on page 174

Advantages of Packages

Easy to upgrade: In the past, when you wanted to install a driver, you installed the driver configuration file. The driver configuration file contained all of the functionality that could be added to a driver. However, there was no easy way to upgrade the configuration file after installation. Packages allow you to upgrade an installed package.

Easy to revert back to factory settings: Packages are easy to install, uninstall, and revert back to a shipping configuration of the driver.

Common functionality can be reused: Functions that are common to the drivers can be grouped in a particular package and the same can be referenced by other drivers. This is not possible with configuration files.

Easy content life cycle management: Managing the life cycle of content is easier with packages due to versioning.

Easy to update: Packages allow you to update the features of a driver without updating the entire driver.

Understanding Package Dependencies

Many packages require one or more other packages to function properly. When you install a package, the package may require other packages also be installed, either as feature sub-packages or separate packages entirely. For example, several packages require you install the default Common Settings package before installing or deploying.

These dependencies are mandatory and are always enforced, indicating a technical dependency one package has for a component of another package.

Understanding Driver Set Packages and Identity Vault Packages

A package can be a **driver package**, a **driver set package**, or an **Identity Vault package**. In general, package dependencies follow a one-way "pyramid" structure. Driver packages can require other driver packages, driver set packages, or Identity Vault packages, and driver set packages can also require Identity Vault packages. However, Identity Vault packages cannot require driver or driver set packages, and driver set packages cannot require driver packages.

Understanding Base Packages and Feature Packages

In addition, a package can be a **base package** or a **feature package**. Feature packages contain the actual functionality a driver uses, broken apart by "feature," while base packages tell Designer how to assemble those feature sub-packages together into an actual driver. Base packages should be used to create a driver and not to deliver content.

Feature packages themselves may be **mandatory** or **optional**, depending on the requirements of the base package. Some features may not be strictly necessary for a driver to function but could be useful for some users, while other features are required for the driver to function properly.

You configure the mandatory and optional feature packages of a base package in the Properties of the base package. When you install a driver, the Driver Configuration Wizard displays both the mandatory and optional features of that driver's base package and installs the mandatory feature packages and allows you to select which optional feature packages you want to install. For more information about configuring mandatory and optional packages, see "Configuring Mandatory and Optional Feature Packages" on page 230

Package Content

Packages are installed on drivers, driver sets, and Identity Vaults. The content of the packages installed on the Identity Vault can affect all of the drivers in the Identity Vault. The content of the packages installed on the driver set can affect all of the drivers in the driver set. The content of the packages installed on a driver only affects that driver.

You can store many different types of objects in a package, including driver objects, library objects, User Application objects, DS object resources, filter extension resources, and package prompt resources. The types of objects you can store in a package depends on the type of the package itself.

NOTE: You can install content on a driver without adding that content to a package, including policies, ECMAScripts, and GCVs. However, if you install content directly on a driver, you cannot control what order the driver runs the content.

For example, if you have a package that contains 10 policies installed on a driver, and one non-package policy also installed on that driver, the non-package policy may run in between two of the package policies, regardless of how you order the policies.

The following table lists the objects the can be installed in the different package types.

Table 6-1 *Package Content in Package Types*

Object Type/Package Type	Driver	Driver Set	Identity Vault
Notification Templates			X
Library		X[1]	X[1]
Credential Application object	X	X[2]	X[2]
Credential Repository object	X	X[2]	X[2]
DirXML Script	X	X[2]	X[2]
ECMAScript	X	X[2]	X[2]
Mapping Table	X	X[2]	X[2]
Global Configuration object	X	X	X[2]
DS object	X	X	X[2]
Resource object	X	X[2]	X[2]
Schema Map	X	X[2]	X[2]
XSLT	X	X[2]	X[2]
Job	X	X	
Entitlement	X		
Entities	X[3]		

Object Type/Package Type	Driver	Driver Set	Identity Vault
Lists	X[3]		
Queries	X[3]		
Relationships	X[3]		
Configuration	X[3]		
Provisioning Request Definitions	X[3]		
Teams	X[3]		
Roles	X[3]		
Role Configuration	X[3]		
Resources	X[3]		
Separation of Duty (SoDs)	X[3]		

[1] Libraries are not packaged, only their contents. Packages store the library's name and location and create it at install time, if it doesn't already exist.

[2] These items can only be added to a package of the respective type if they are in a library.

[3] These items can only be added to a User Application driver package.

Installing or Upgrading Packages

Use the following list of tasks to install, add, upgrade, or import packages. For information about creating or copying packages, see “Developing Packages” on page 193.

- “Installing Packages” on page 175
- “Adding Packages” on page 178
- “Upgrading Installed Packages” on page 179
- “Importing Packages into the Package Catalog” on page 180
- “Managing Installed Packages” on page 181

Installing Packages

You can install packages on Identity Vaults, on driver sets, or on drivers. You can verify the packages have been imported by following the instructions in “Importing Packages into the Package Catalog” on page 180.

There are three different types of packages based on the package are installation target: **Identity vault packages**, **driver set packages**, and **driver packages**.

Driver packages are further grouped as:

- **Driver Base Configuration Packages:** Contains the base functionality for a driver. You must install a driver base configuration package first.
- **Mandatory Features Packages:** If there is a feature that is required for a driver to function, but is not included in the driver base configuration package, it is added to a mandatory features package.
- **Optional Features Packages:** Contains features for a driver that aren't mandatory for the driver to function.

To install packages on an existing Identity Vault, driver set, or driver, see "Adding Packages" on page 178.

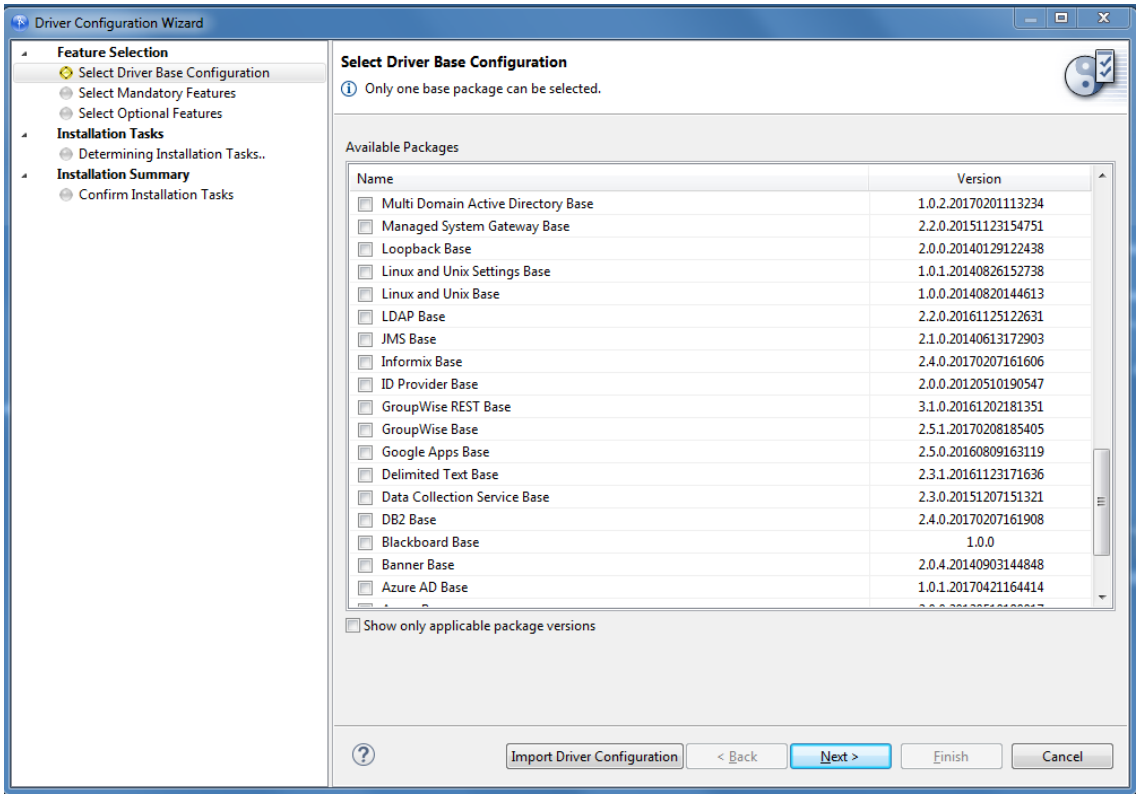
To install a new driver, including the packages that make up the driver, use the following procedure:

1. Drag and drop an application from the Palette into the Modeler.

   or

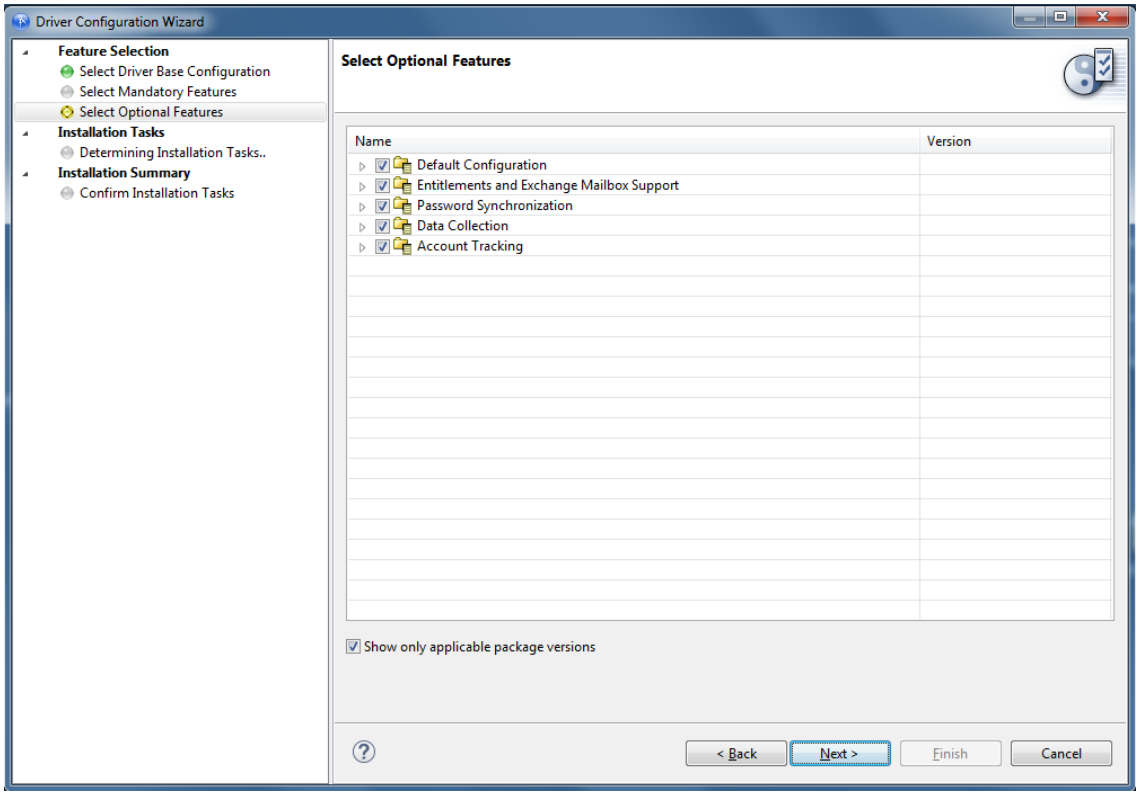
   Right-click the driver set in either the Outline view or in the Modeler, then click **New** > **Driver**.
2. Click the check box next to the base package you want install, then click **Next**.

   NOTE: You can only install one base package per driver.



3. (Conditional) If you want to install any of the available optional features for the base package you selected, ensure the check box next to those packages is selected. Most options are selected by default because they are recommend for the driver.

NOTE: In most installations, you are recommended to install all optional features.



Optional packages are grouped by feature. You can expand features to see the specific packages installed for each. You must select a feature to install the packages for that feature.

4 (Conditional) If you do not want to install a particular optional feature, clear the check box for that package.

5 Click **Next**.

6 (Conditional) If the base package requires a dependent package, Designer prompts you to install the dependent package. Select the dependent package, then click **OK**.

7 Respond to any prompts, if necessary, then click **Next**.

The prompts are specific for each driver. Each driver guide contains the specific instructions for that driver. See the OpenText Identity Manager driver documentation Website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/) for the specific driver information.

8 Review the installation summary, then click **Finish**.

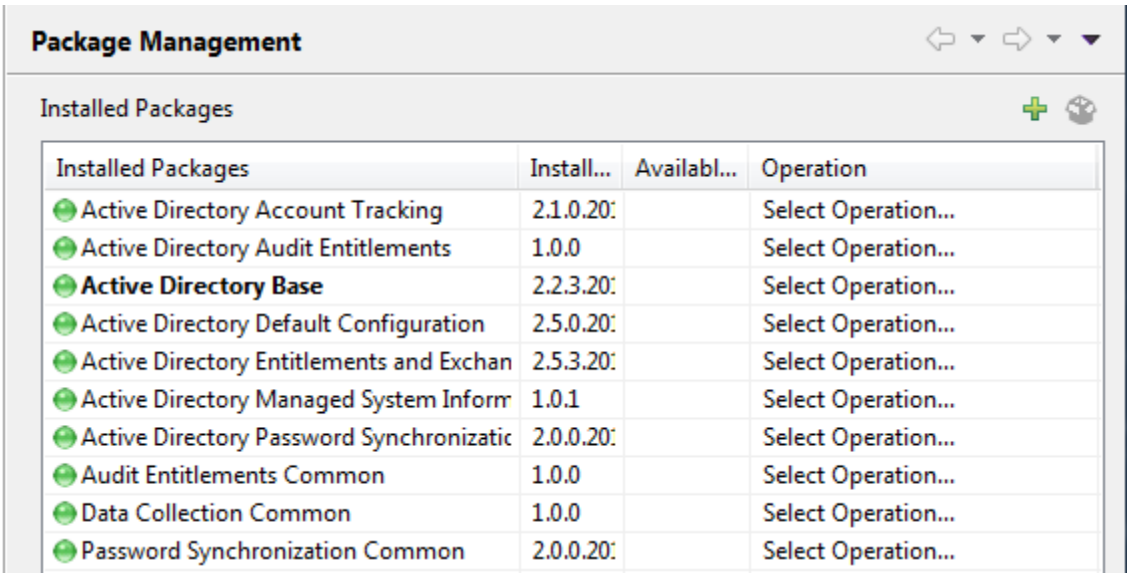
After the packages are installed, the driver contains the functionality included in the packages.

Adding Packages

You can add new functionality to an existing driver by adding new packages to an existing Identity Vault, driver set, or driver.

1. Right-click the Identity Vault, driver set, or driver, then click **Driver > Properties**.
2. Click **Packages**, then click the **Add Packages** icon .
3. Select the packages to install. If the list is empty, there are no available packages to install.
4. (Optional) Deselect the **Show only applicable package versions** option, if you want to see all available packages.

   This option is only displayed on drivers. By default, only the packages that can be installed on the selected driver are displayed.
5. Click **OK**.
6. Click **Apply** to install all of the packages listed with the Install operation.



7. (Conditional) Fill in the prompts with appropriate information to install the package, then click **Next**.

   Depending on which package you selected to install, you might have fields that you must fill in. For detailed information about the fields, see the specific driver guide at the OpenText Identity Manager drivers documentation Website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/).
8. Read the summary of the installation, then click **Finish**.
9. Click **OK** to close the Package Management page after you have reviewed the installed packages.
10. Repeat Step 1 through Step 9 for each Identity Vault, driver set, and driver where you want to add the new packages.

Upgrading Installed Packages

Designer provides the ability to upgrade the installed packages in two ways:

- "Using the Package Upgrade Method from the Properties Page" on page 179
- "Using the Package Upgrade Consolidated View" on page 179

Using the Package Upgrade Method from the Properties Page

You can upgrade any package that is installed in your OpenText Identity Manager environment if there is a newer version of the package available.

Complete the following steps to upgrade an installed package:

1. Ensure you add any GCVs included in the package to a new GCV Resource object. For more information, see the "Global Configuration Value Definition Editor" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.
2. Right-click the Identity Vault, driver set, or driver where the package is installed that you want to upgrade, then click **Driver > Properties**.
3. Click **Packages**.

   If there is a newer version of a package, there is check mark displayed in the Upgrades column.
4. Click **Select Operation** for the package that indicates there is an upgrade available.
5. From the drop-down list, click **Upgrade**.
6. Select the version that you want to upgrade to, then click **OK**.

   NOTE: Designer lists all versions available for upgrade.

   ---
7. Click **Apply**.
8. (Conditional) Fill in the fields with appropriate information to upgrade the package, then click **Next**.

   Depending on which package you selected to upgrade, you might have fields that you must fill in to upgrade the package. For detailed information about the fields, see the specific driver documentation located on the OpenText Identity Manager drivers documentation website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/).
9. Read the summary of the installation, then click **Finish**.
10. Review the upgraded package, then click **OK** to close the Package Management page.

Using the Package Upgrade Consolidated View

The Package Upgrade Consolidated View helps you to view and upgrade the active packages of your entire OpenText Identity Manager project in a single view to the latest available versions. The provision of a single view removes the need for you to separately go to each driver, driver set, or Identity Vault in your project to view or perform an upgrade on the packages they contain.

Complete the following steps to upgrade an installed package:

1. In the Outline view, right-click **Package Catalog**, then select **Package Upgrade**.

   The Package Upgrade Consolidated View dialog box appears.

2 Select one or more packages from the list, then click **OK**.

If your OpenText Identity Manager environment is already up-to-date, the list is empty. Otherwise, select the packages that you want to upgrade.

In addition, the Project Checker displays the list of available packages upgrades in your Identity Management environment.

Importing Packages into the Package Catalog

Designer adds packages to the Package Catalog dynamically. However, if you need to add a custom package to the Package Catalog, you can import the package .jar file.

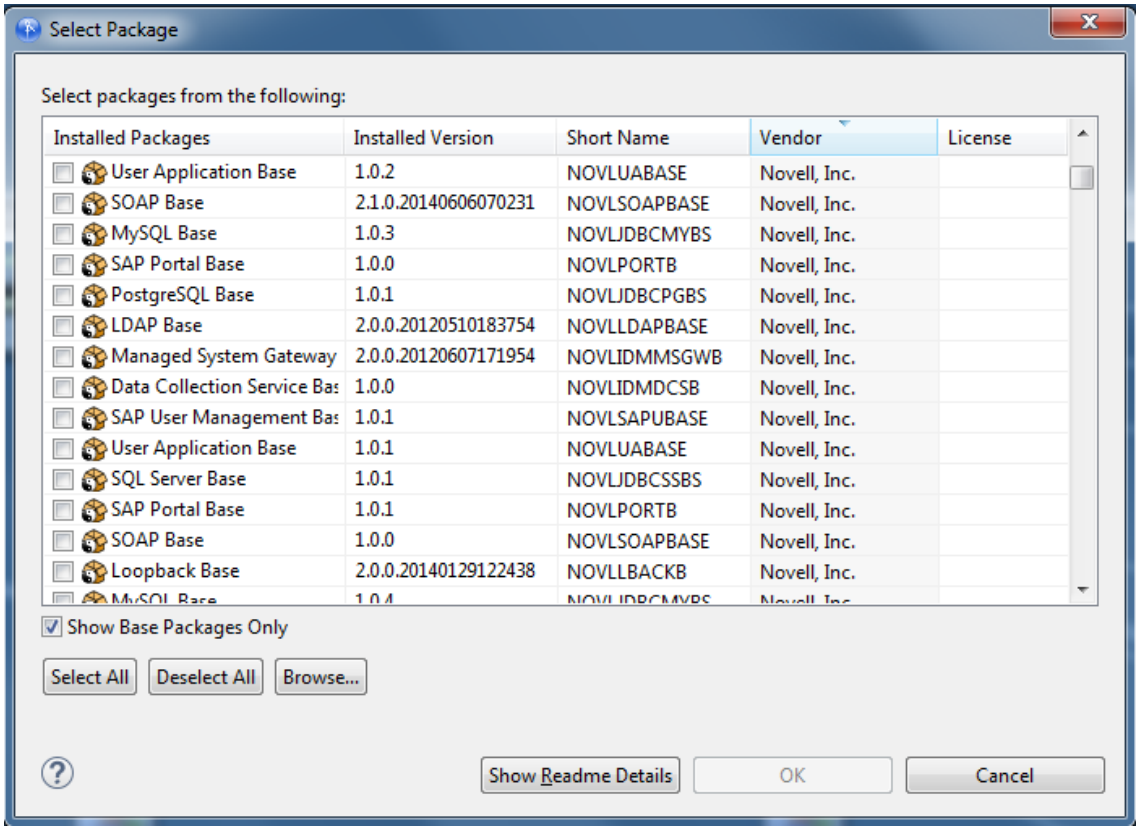
Use the following procedure to import one or more packages into the package catalog.

1 In the Outline view, right-click **Package Catalog**, then select **Import Package**.

2 Select one or more packages from the list. If all of the available packages are already imported, the list is empty.

or

Click **Browse**, then browse to and select a package on the file system and click **OK**.



3 Click **OK** to import the selected packages.

4 Review the import message, then click **OK**.

After you import a package, you must install the package on a driver before you can use that package. Continue with "Installing Packages" on page 175 for instructions.

Managing Installed Packages

The **Manage Packages** option allows you to manage an installed package in your OpenText Identity Manager environment. This ensures that you only have the required packages in your environment, thus improving the performance of the system.

The Manage Package window is displayed when Designer is launched for the first time. Alternatively, navigate to **Help > Manage Packages**.

Let us understand the usage of the Manage Package feature through an example. For example, perform the following steps to manage an installed package for the User Application driver:

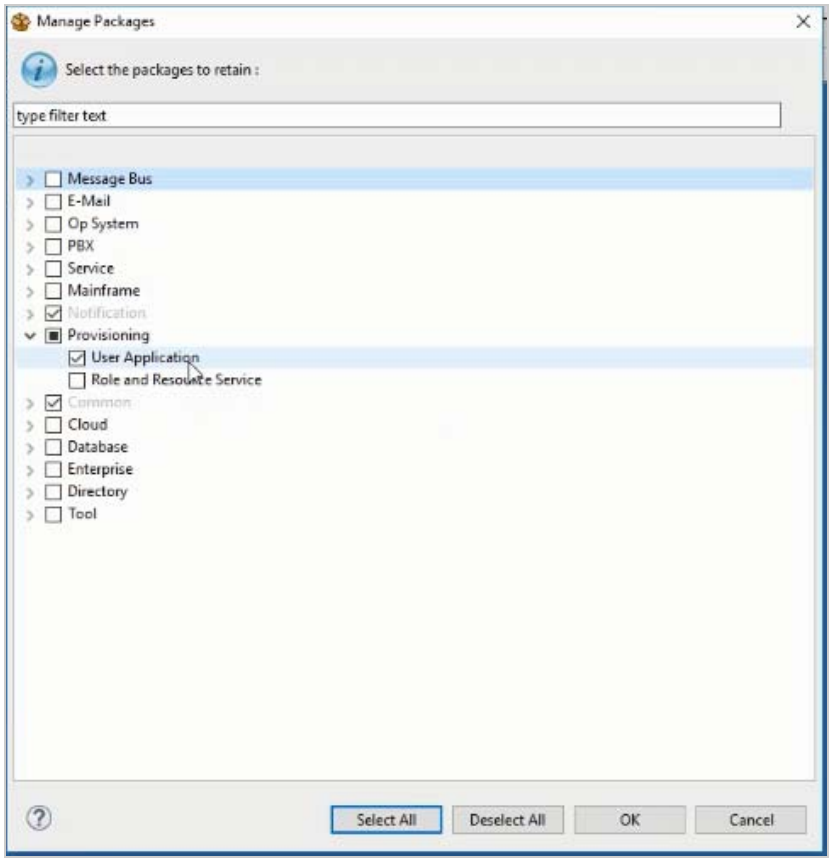
1 Navigate to **Help > Manage Packages**.

In the **Manage Packages** window, select the packages you wish to retain.

NOTE: ◆The **Notification** and **Common** packages are selected by default. You cannot de-select these packages.

- The **OK** button is enabled only after you select a package in addition to the packages selected by default.

Figure 6-2 *Select Provisioning > User Application*



The selected packages are saved in the `<Designer installed location>/packages/eclipse/plugin` folder.

By default, all the dependent (mandatory and optional) packages for the User Application driver that are already deployed to the Identity Vault are retained.

IMPORTANT: ◆Administrators who use Designer to upgrade or downgrade any driver must retain all the listed packages.

- Consultants working on any specific package(s), must retain only those packages.
- If you delete a package, the deleted package specific to the driver continues to work in that project using the local copy.
- The driver will not work in any other project, even if it is in the same workspace. You cannot upgrade or downgrade a package in a working project.

---

2 Click **OK**.

3 Click **Yes** on the confirmation dialogue message.

4 Click **Yes** to restart Designer for the package retention changes to take effect.

After Designer restarts, a pop-up window displays stating that the selected packages are successfully retained.

5 Click **Yes**.

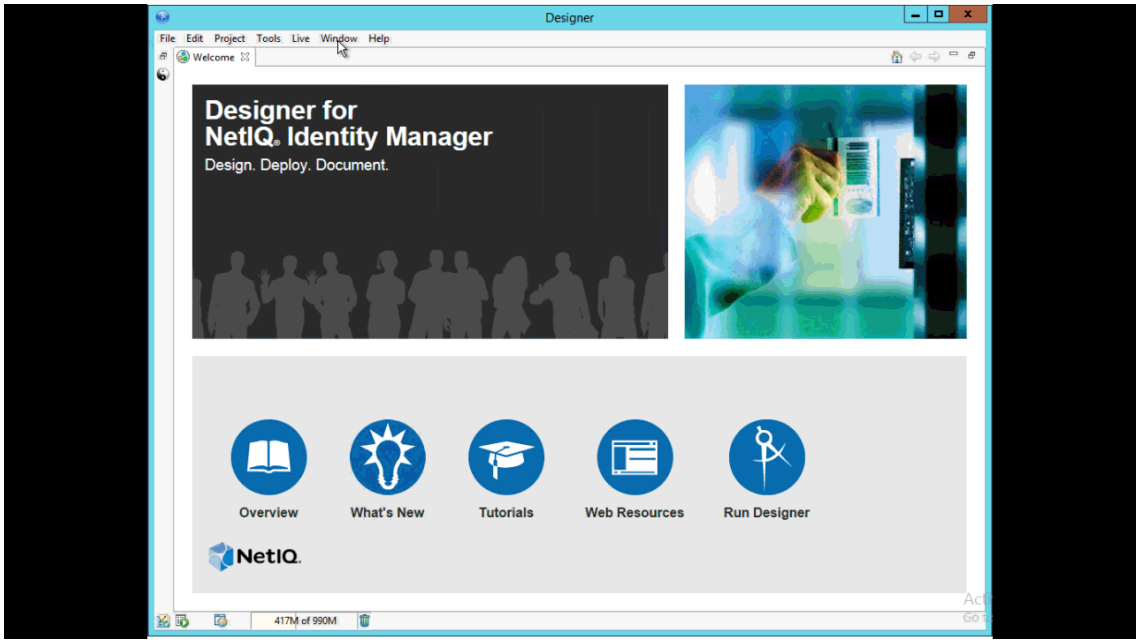
6 (Conditional) To view the retained packages and their dependent packages, navigate to **Help > Manage Packages**. You will notice that only the User Application package is retained.

To reinstall the other packages, perform an online update and then check for package updates. For more information, see "Including Existing Packages to Drivers" on page 182.

Including Existing Packages to Drivers

To include existing packages:

1 Navigate to **Windows > Preferences > NetIQ > Package Manager > Online Updates** and ensure the **Local Offline** site option is selected.



2 Click **Apply** and **OK**.

3 Navigate to **Help > Check for package updates**, select the package(s) you want to import.

NOTE: The **Packages_backup** folder contains only those packages that were shipped with the original version of Designer that was installed. Any packages that have been updated later are not stored in this folder. For latest package updates, perform an online update through the Preferences option. For more information, see "Online Updates".

Customizing Default Packages

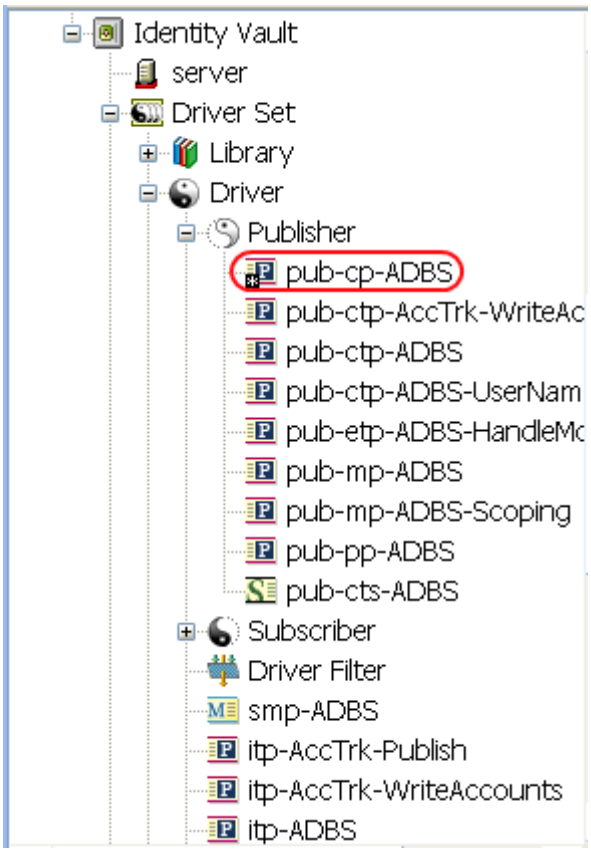
In most cases, when you install a default package shipped by OpenText in your environment, you need to customize that package for the driver to function properly. You may need to add new policies to the default package, modify existing policies and filter extensions, and configure schema mapping policies for your environment. You can modify the content of a default package at any time using tools provided in Designer, like the Policy Builder.

For more information about creating or modifying policies, see "Managing Policies with the Policy Builder" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*. For more information about modifying filters, see "Controlling the Flow of Objects with the Filter" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*. For more information about configuring schema mapping policies, see "Defining Schema Map Policies" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

NOTE: If you have previously worked with driver configuration files, note that there are no additional steps required to make changes to the package content. You use Designer as you would in the past to change a policy, filter, or any other object that is delivered in a package.

Each package has a checksum file, so that when you make changes to the content delivered in the packages, Designer keeps track of those changes. Designer adds an icon to content that is customized. In the figure below, the pub-cp-ADBS policy has changed, where all of the other policies have not changed since the package was installed.

Figure 6-3 *Changed Policy*



If there is a new package available and you have customized the package, Designer prompts you to keep your changes or overwrite the customization with the new package content.

You can also revert the customization that you made to any package at anytime.

1. In the Outline view, select an object that has changed.
2. Right-click the selected object, then click **Revert Customization**.

The content is reverted back to the state it was when the package was first installed. The Revert Customization option is like an Undo option.

Managing Package Versions Using Git

We allows you to manage Designer package versions using Git. Git is an open source distributed version control system used for storing and retrieving files. It allows you to manage, track, maintain the history of changes, or retrieve an earlier state or compare different states of files. For more information about Git, see Git documentation.

IMPORTANT: You can use Git to share packages when the packages are developed or staged. We do not recommend you to use Git for Designer project versioning.

Benefits of Using Git

Implicit backup: Git maintains multiple backup copies within the repository. This is useful in case of an event or disk crash.

Pull and Push operation: Git allows you to copy changes to a remote or local repository.

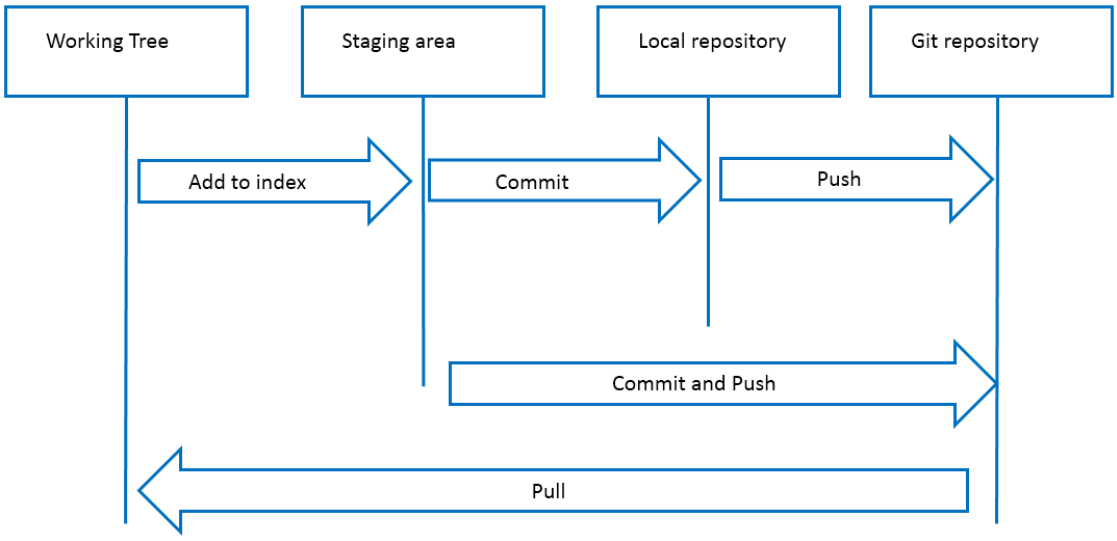
- The **Pull** operation fetches the changes from the Git repository to a local repository. This is useful when you want to synchronize changes between the two repository instances.
- The **Push** or **Publish** operation publishes the changes from the staging area to a local repository and further publish them to the Git repository. You can also publish your changes directly from the staging area to the Git repository.

How Does Git Work?

You must create a source Git repository and add the required files to it. Designer pulls files from the Git repository. The changes are first performed on the **Working Tree**. Then the changes are staged in the **Staging Area** before they are pushed to the Git repository.

- **Working Tree:** This consists of the files that you are currently working on. You can add, modify, or delete a file from the local code base location. You can modify multiple files at a time. You add the changes to the **Working Tree** after modifying the required information.
- **Staging Area:** This is an intermediate storage area. This area has the changes that were made to the files in the **Working Tree**. Git collects all changes which will be part of the next commit action from this area.
- **Git Repository:** This is the area that stores all the committed changes. The final version of the changes are available in the Git repository.

Figure 6-4 *Commit, Push, and Pull operation*



Let us understand how Git works. You first edit the files in the working tree. After editing the file, you move the file from the working tree to the staging area. The **Commit** option is used to commit the changes from the staging areas to the local repositories. You use the **Push** option again to commit changes from the local repository to the Git repository.

You can directly commit the changes from the staging area to the local and Git repository by using the **Commit and push** option.

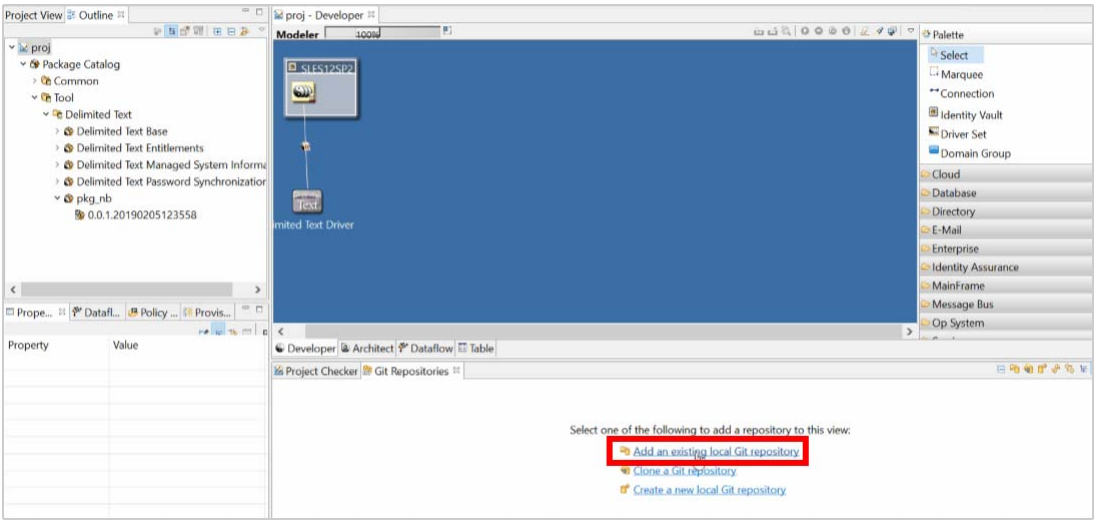
NOTE: Eclipse supports only EGit and JGit plug-ins.

Adding an Existing Local Git Repository

Perform the following actions to add an existing local Git repository:

1. Navigate to **Windows > Show View > Other > Git > Git Repositories**.
2. Click **Add an exiting local Git repository**.

Figure 6-5 *Add an existing local Git repository*



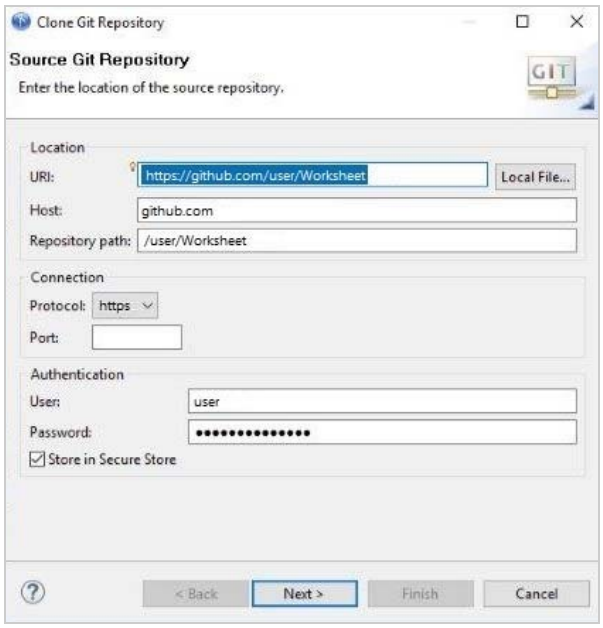
3. Select the required repository and click **Finish**.

Cloning an Existing Local Git Repository

Perform the following actions to clone an existing Git repository:

1. Navigate to **Windows > Show View > Other > Git > Git Repositories**.
2. Click **Clone a Git repository**.
3. Specify the Git repository URL you wish to clone in the URI field in order to connect it to the local repository. When the URL is entered, the remaining fields are auto-populated. Enter the Git credentials for authentication.

Figure 6-6 *Clone Git Repository*



4 Click **Next**.

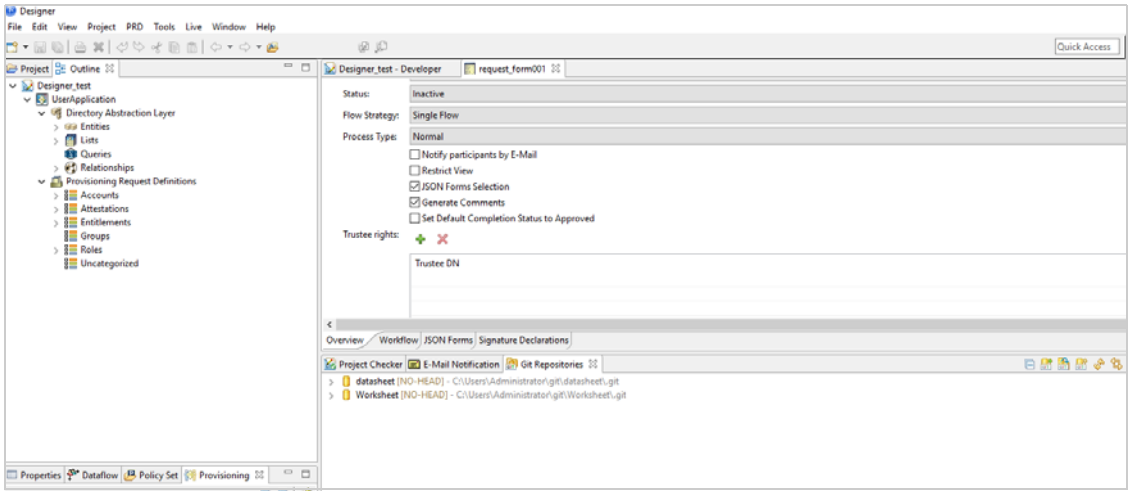
5 (Optional) Select the branch (if any) in the Branch selection window and click **Next**.

By default, a local folder is created with the same repository name. This directory is an editable field. Click **Finish**.

NOTE: Select the **Run in Background** option to allow the current operation to execute in parallel with other Designer activities.

The cloned repository is now listed under **Git Repositories**.

Figure 6-7 *Cloned repository listed under Git Repositories*



6 In the **Outline view**, right click on the package and select **Build**.

7 Select the **Build to cloned Git Repository**; and select **Release Package** if you wish to release the package.

NOTE: The **Build directory** option is grayed out when Build to cloned Git repository is selected.

For more information about releasing package, see "Releasing and Publishing Packages" on page 240.

8 Click **OK**.

9 A success message appears. Click **OK**.

The built package is listed under **Working Tree**.

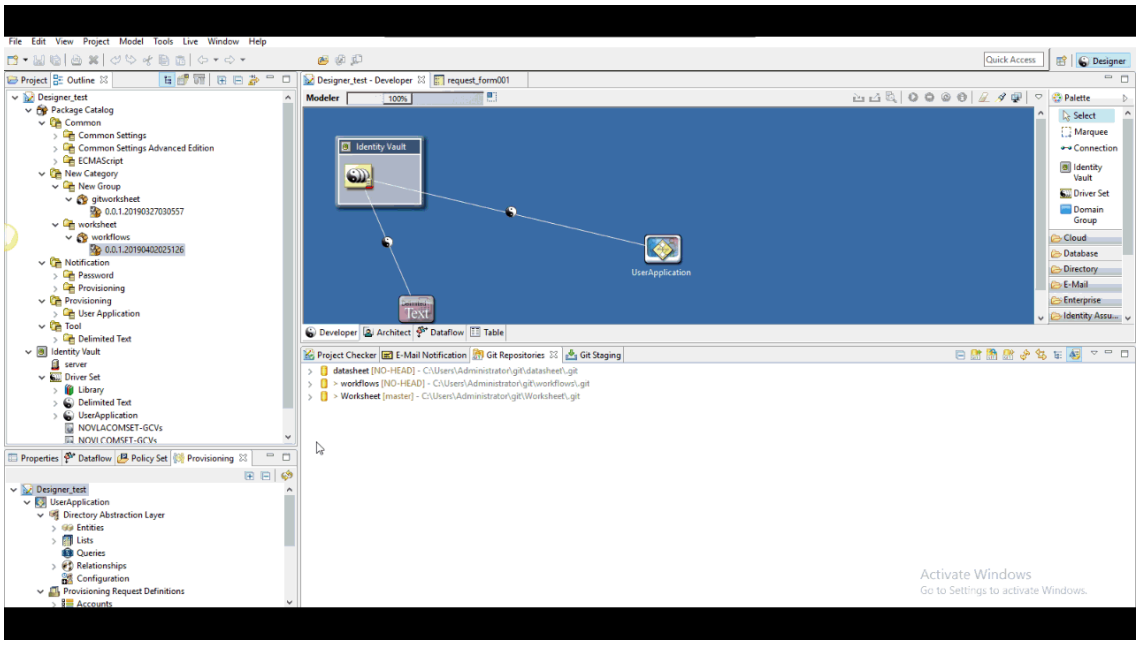
NOTE: Each time you edit and build the package, a new version of the package is appended to the working tree.

10 Right click the cloned repository and click **Commit**. The **Git Staging** tab appears.

11 In the **Git Staging** tab, the package is listed under **Unstaged Changes**. Right-click on the package and select **Add to Index**. The package will be listed under **Staged Changes**.

NOTE: You must enter a message while committing your changes to the Git repository.

12 To check-in the changes only to the local repository, click **Commit**. Click **Commit and Push** for the changes to be checked-in to both the local and Git repository.



13 Click **Next**.

14 Click **Finish**.

The package is listed in the Git repository.

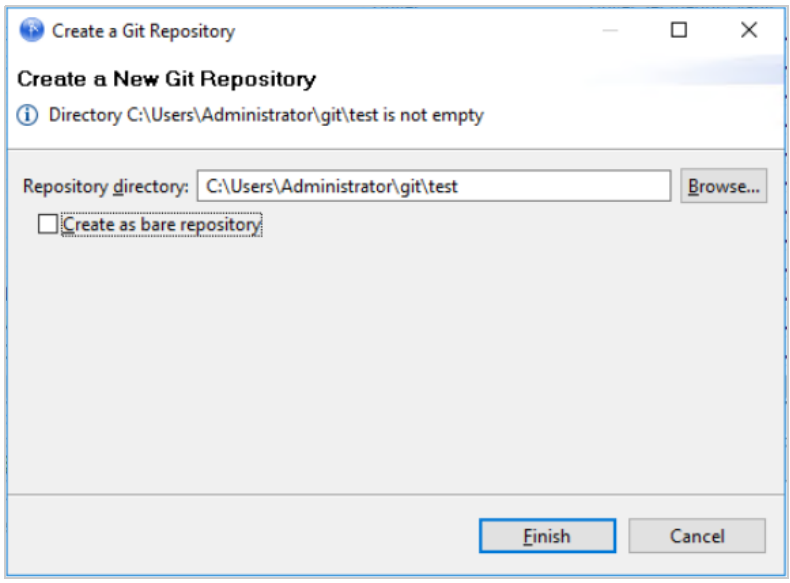
For more information about publishing the package, see "Releasing and Publishing Packages" on page 240.

Creating a Local Git Repository

Perform the following actions to create a new local Git repository:

1. Navigate to **Windows > Show View > Other > Git > Git Repositories**.
2. Click **Create a new local Git repository**.
3. Specify the directory for the new repository. By default, a local folder is created. This is an editable field and click **Finish**.

Figure 6-8 *Create new local Git repository*



When you add files to the local GIT repository, the `.gitignore` file blocks some of .jar files by default. To manually add ignored .jar files to the repository, run the command:

```
git add --force <file>
```

For example, `git add --force NAME_Of_THE_FILE.jar`.

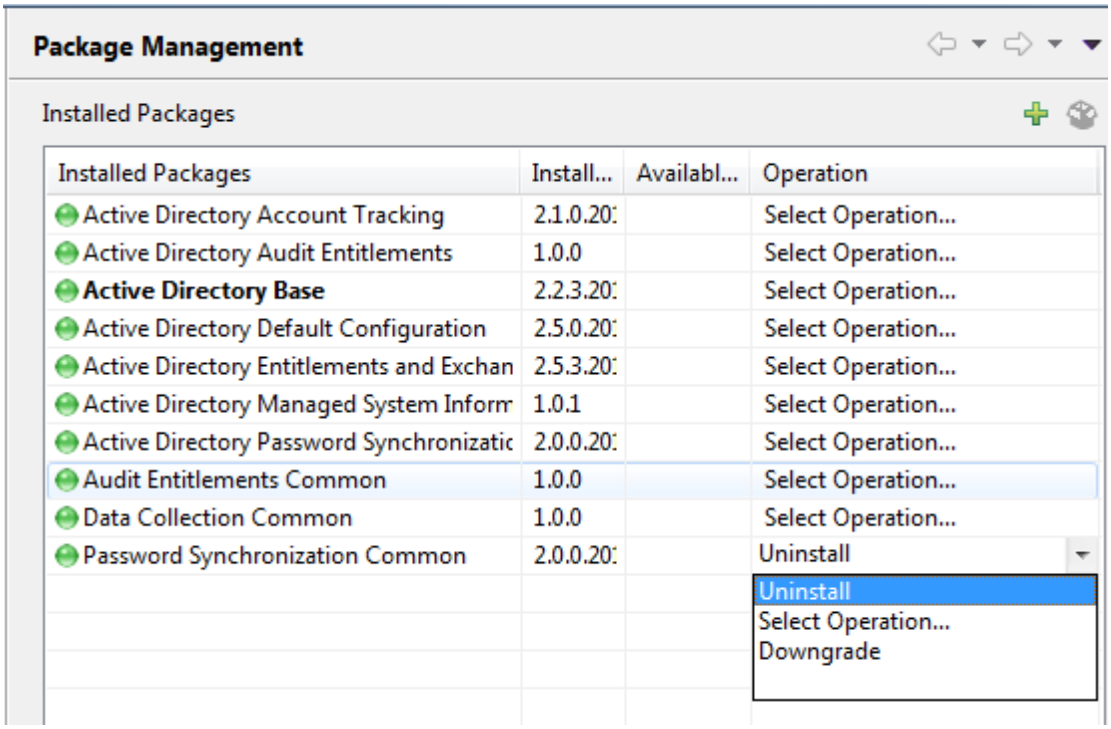
Removing or Downgrading Packages

Use the following list of tasks to remove, uninstall, or downgrade packages or to enable or disable factory mode on a driver.

- "Uninstalling Packages" on page 190
- "Downgrading Installed Packages" on page 190
- "Removing Packages from the Package Catalog" on page 191
- "Running a Driver in Factory Mode" on page 191
- "De-activating Factory Mode" on page 192

Uninstalling Packages

1. Right-click the Identity Vault, driver set, or driver where the package is installed that you want to uninstall, then click **Properties**.
2. Click **Packages**, select the package you want to uninstall, then click the **Select Operation** cell.
3. Click **Uninstall** from the drop-down list.



4. Click **Apply** to uninstall the package, then click **OK** to close the Package Management page.

Downgrading Installed Packages

You can downgrade any package that you have upgraded. This allows you to revert the driver back to a known state for troubleshooting purposes.

1. (Optional) Before downgrading an installed package, you may want to create a backup of all of the customized policies in the package. For information about backing up drivers in OpenText Identity Manager, see *OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Linux* or *OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Windows*.
2. Right-click the Identity Vault, driver set, or driver where the package is installed that you want to downgrade, then click **Properties**.
3. Click **Packages**, then click the **Select Operation** option for the package you want to downgrade.
4. From the drop-down list, select **Downgrade**.
5. Select the version that you want to downgrade to, then click **OK**.

All versions that are available to downgrade to are listed.

6 Click **Apply**, then click **Finish** to downgrade the package.

Removing Packages from the Package Catalog

You can remove unused packages from the package catalog all at once or delete a specific package if the package is currently not in use. If you try to delete a package that is in use, you get an error message.

If you want to remove all unused packages from the package catalog, complete the following steps:

1 Right-click the package catalog and select **Remove Unused Packages**.

2 Review the list of packages to be removed and click **OK**.

If you want to delete a specific package from the package catalog, complete the following steps:

1 Verify that the package is currently not installed:

1a Right-click the package in the package catalog, then click **Properties**.

1b Click **Targets**.

This page lists all of the objects where the package is currently installed in your project.

1c Click **OK** to close this page.

1d If the package is currently installed, follow the instructions in "Uninstalling Packages" on page 190 to uninstall the package. After the package is uninstalled, continue with this procedure.

2 Right-click the package in the package catalog, then click **Delete**.

3 Click **Yes** to confirm.

Running a Driver in Factory Mode

Designer also provides an option to remove any customizations from a driver while retaining package configuration values and parameters. Customizations can include policies, GCVs, and package prompts.

To run the driver without customizations is called Factory mode. The Factory mode allows you to remove customizations from the driver through one procedure instead of removing customizations from each package.

Factory mode is most useful for package developers who create their own custom packages for use by other users. If you develop a package for a customer, and the customer encounters problems with the driver after installing the package, you can enable Factory mode to troubleshoot those problems on a "clean" driver.

NOTE: ◆We do not recommend enabling Factory mode for shipped drivers or packages, as the default drivers provided by OpenText require customization to work in your environment.

- You can only enable Factory mode on an individual driver. You cannot enable Factory mode on an Identity Vault or driver set.
- Enabling Factory mode affects all driver content, including all pre-configured and custom packages installed on the driver.

There are two options for using Factory mode:

- **Strict:** Removes all customizations and custom configurations from your driver. Custom configurations are new policies, jobs, mapping policies, or other objects created on the driver.
- **Relaxed:** Removes all customizations but no custom configurations from your driver.

To run a driver in Factory mode:

1. In the Outline view or in the Modeler, right-click the driver, then click **Driver > Properties**.
2. Click **Packages**, then select **Run driver in Factory mode**.
3. Select how Package Manager handles the customizations and custom configuration of your driver. You can select either **Strict** or **Relaxed**.
4. Click **Activate** to save the selected change.
5. (Optional) Click the **Configure Factory mode** icon if you want to change the selected option, then click **Activate** again.
6. Click **Apply** or **OK** to make the change active.

De-activating Factory Mode

When you turn off Factory mode on the driver, Package Manager does the following:

- Restores all package customizations, including policies, GCVs, and package prompts
- Restores custom configurations, if you selected Strict
- Preserves package configuration values and parameters

To de-activate Factory mode:

1. In the Outline view or in the Modeler, right-click the driver, then click **Properties**.
2. Click **Packages**, then deselect **Run driver in Factory mode**.
3. (Optional) Select **Reset driver** to permanently reset the driver to factory defaults. When you select this option, the following tasks are performed:
   - All package customizations are deleted
   - Custom configuration are deleted (only if you are in strict mode)
   - Package configuration values and parameters are preserved
4. (Optional) Select **Save driver configuration** to create a driver configuration file that contains the currently values, parameters, and customization.
5. Click **De-Activate**.
6. Click **Apply** or **OK** to make the change active.

7 Developing Packages

In addition to working with and modifying the default set of packages included in Designer, you can create your own custom packages tailored to your particular environment.

- "Why Use Custom Packages?" on page 193
- "Developing Custom Packages" on page 194
- "Preparing to Develop Packages" on page 195
- "Creating a Base Package" on page 197
- "Configuring Initial Settings" on page 199
- "Working with Package Prompts" on page 204
- "Creating Identity Vault and Driver Set Packages" on page 226
- "Creating Feature Packages" on page 229
- "Configuring Mandatory and Optional Feature Packages" on page 230
- "Adding Content to Packages" on page 231
- "Copying Packages" on page 235
- "Building Packages" on page 237
- "Versioning Packages" on page 237
- "Localizing Packages" on page 238
- "Adding and Configuring Licenses" on page 239
- "Releasing and Publishing Packages" on page 240
- "Best Practices for Package Development" on page 241

Why Use Custom Packages?

For many users, the default set of packages you can install with Designer addresses all the relevant areas of their OpenText Identity Manager environment.

However, at some point you may need to create a custom package outside of the default packages provided by OpenText. You might need to modify a shipped package, copy a shipped package, modify and rebrand that package for use in your environment, or create a completely new package for a custom driver.

The following sections help you to create a custom package.

Developing Custom Packages

Creating custom packages involves a different set of tasks than managing packages. You can create packages for Identity Vaults, driver sets, and drivers. You can develop custom packages by completing the following steps.

Before you start developing custom packages, we recommend that you also read "Best Practices for Package Development" on page 241.

Steps	See Section
1. **Configure default package preferences in your Designer environment.**	"Setting Default Package Preferences" on page 195
2. **Create a development driver.**	"Creating a Development Driver" on page 195
3. **Enable package development mode.**	"Enabling Package Development Mode" on page 196
4. **Define the overall package structure.**	"Defining Custom Package Structure" on page 196
5. **Create a custom base package.**	"Creating a Base Package" on page 197
6. **Configure initial settings for the base package and sub-packages.**	"Configuring Initial Settings" on page 199
7. **Add package prompts to the base package.**	"Working with Package Prompts" on page 204
8. **Create common Identity Vault and driver set packages.**	"Creating Identity Vault and Driver Set Packages" on page 226
9. **(Optional) Add libraries to Identity Vault and driver set packages.**	"Creating Libraries" on page 227
10. **(Optional) Add GCVs to Identity Vault and driver set packages.**	"Adding GCV Resource Objects" on page 228
11. **(Optional) Add notification templates to Identity Vault and driver set packages.**	"Adding Notification Templates" on page 228
12. **Create custom feature packages.**	"Creating Feature Packages" on page 229
13. **Configure mandatory and optional feature packages.**	"Configuring Mandatory and Optional Feature Packages" on page 230
14. **(Optional) Add GCV resources to feature packages.**	"Adding GCVs to Feature Packages" on page 232
15. **(Optional) Add package prompt resources to feature packages.**	"Adding Prompt Resources" on page 232
16. **(Optional) Add policies to feature packages.**	"Adding Policies" on page 233
17. **(Optional) Add filter extensions to feature packages.**	"Adding Filter Extensions" on page 234
18. **(Optional) Copy an existing package, if necessary.**	"Copying Packages" on page 235
19. **Build and test your custom packages.**	"Building Packages" on page 237

Steps	See Section
20. **If previous versions of your packages exist, update the version.**	“Versioning Packages” on page 237
21. **(Optional) Export strings and prompts from your packages and send for localization.**	“Localizing Packages” on page 238
22. **(Optional) Release and publish your custom packages for other users to download and install.**	“Releasing and Publishing Packages” on page 240

Preparing to Develop Packages

The first step in developing custom packages is to prepare your Designer environment. You should create a new Designer project, install a valid Identity Vault, configure any default preferences, create a development driver to use as an installation target, enable package development mode, and define the overall structure for your packages.

For more information about creating a project, see “Creating a Project” on page 19. For more information about installing an Identity Vault, see “Creating a Model” on page 23.

Setting Default Package Preferences

Before you start creating custom packages, we recommend you configure default Package Manager preferences as necessary in your environment. In particular, you should configure your **Vendor Defaults**, **License Defaults**, and **Locations Defaults** preferences.

To configure your preferences, click **Window > Preferences**, then expand **NetIQ > Package Manager** and modify preferences as necessary. For more information about preferences in Designer, see “Setting Preferences” on page 479.

Creating a Development Driver

Complete the following steps to create a “blank” development driver you can use as a target for your custom packages.

1. Drag and drop an application from palette into the Modeler to launch the Package Installation Wizard. The application can be of any type.

   NOTE: The Package Installation Wizard does not show any packages if the catalog is empty.

2. When Designer displays the Driver Configuration Wizard, click **Cancel**, without installing or configuring any packages. Designer creates an empty driver in the Modeler and links the driver to your Identity Vault. You can then use to add your own custom content.

Enabling Package Development Mode

Packages can only be created and modified when the Identity Vault is running in package development mode.

1. Either in the Outline view or the Modeler, right-click the Identity Vault, then click **Properties**.
2. Select **Enable Package Developer Mode**, then click **OK**.

NOTE: If you disable package development mode, you can then only view the properties of a package in the Package Catalog or compare the current version of a package to other available package versions.

You cannot create packages, add objects to packages, remove objects from packages, or sync packages on a driver or driver set with package development mode disabled.

Defining Custom Package Structure

At the start of the package-creation process, you should define the structure you want to use for the packages you create, including mapping out the specific base packages and feature packages you need.

Use questions like the following to define your package structure:

- To which package categories and groups will your packages belong?
- To which driver types does this package apply?
- On which targets do you plan to install packages?
- Which feature packages are mandatory?
- Which feature packages are optional?
- Which features can be used by other drivers?
- Which package prompts or settings will be used across feature packages and need to be stored in a base package?
- Does your package or driver require functionality included in any default packages?
- Can some functionality be included in higher-level driver set and Identity Vault packages, for use by all packages and drivers?

In addition to creating new prompts, GCVs, and other objects, you can use the “common” packages provided by OpenText in your own package or driver.

For example, the Common Settings (NOVLCOMSET) driver set package configures the default location for storing user and group identity information in the Identity Vault, and the default LDAP Classes (NOVLLIBLDAP) driver set package includes an ECMAScript that allows you to search any LDAP source from OpenText Identity Manager. Before developing your own custom packages, we recommend you familiarize yourself with the existing functionality provided in the default packages.

For information about configuring mandatory and optional packages, see “Configuring Mandatory and Optional Feature Packages” on page 230. For best practice information about configuring package dependencies, see “Defining Package Relationships” on page 242.

Creating a Base Package

When creating custom packages, you first need to create a new base package. The base package acts as a master list that tells Designer how to assemble all the custom sub-packages you create.

Base packages should not contain content such as policies or resource objects. We recommend only including package prompts and initial settings information in your base package.

WARNING: Designer does not automatically check if a package functions properly or is complete. If you attempt to deploy a package that is incomplete or does not work correctly, you can inadvertently modify your package targets.

Complete the following steps to create a custom base package.

1. (Optional) If you want to create a new package category, navigate to the Outline view in Designer and complete the following steps:
   - **1a** Right-click the package catalog, and then select **New Category**.
   - **1b** Specify the name of the category, then click **OK**.

     For example, if you want to create a base package for a database application driver, you could specify `Database` as the category name.
2. (Optional) If you want to create a new package group within a category, complete the following steps:
   - **2a** Right-click the package category where you want to create a group and select **New Group**.
   - **2b** Specify the name of the package group, then click **OK**.

     For example, if you want to create a base package for a database application driver, you could specify the name of the specific database application as the group name.
3. Right-click the package group where you want to create a new package and select **New Package**.

   NOTE: All packages must belong to a category and a group within that category. You cannot create a package outside of a package group.

   ---
4. Specify a name, version number, and description for the package in the appropriate fields.
5. Specify a short name for the package in the appropriate field. OpenText Identity Manager and Designer display the specified short name when you open the package in a user interface. This name must be unique in the Identity Vault.

   NOTE: The standard short name for a package is 12 characters long, separated into three sections of four characters: `[Vendor][Target system][What package does]`.

   For example, if you have a base Active Directory package created by OpenText, the package short name could be `NTIQADIRBASE`.

   ---
6. Click the **Type** drop-down menu and select **Driver**.
7. Select **Base Package**.
8. Verify the package category and group are correct.
9. Click **Next**.

10 In the **IDM Compatibility** section, select the minimum and maximum versions of OpenText Identity Manager that this package is compatible with. For example, if you create a new package in an Identity Manager 4.7 environment that uses a feature only available in 4.5, you can use the minimum version to prevent users with Identity Manager 4.0.1 or earlier to install the package.

11 In the **Application Compatibility** section, select the minimum and maximum versions of the managed application that this package is compatible with.

NOTE: OpenText Identity Manager does not currently enforce restrictions on the minimum and maximum application versions specified. OpenText Identity Manager can only provide a recommendation to user who try to install the package.

---

12 Select one or more driver types in the **Available Driver Types** list with which you want the package to be compatible and use the right-arrow icon to move them to the **Supported Driver Types** list

---

NOTE: The package must support at least one driver type. Ensure you select the type of application you used when creating your development driver, or select <All> if you want the package to support all possible driver types.

---

13 Click **Next**.

14 Specify or modify the vendor information you want to include in the package, then click **Next**. You must specify the vendor name for the package.

15 Review the Summary page and click **Finish**.

16 (Optional) If you want to require a particular non-feature package, like a common driver set package, be installed along with your base package, complete the following steps:

16a In the Outline window, expand the Package Catalog and navigate to the version of the base package you created in the preceding steps.

16b Right-click the base package and select **Properties**.

16c In the Properties window, click **Dependencies**.

16d Click the plus icon to and select the package you want to add as a dependency. For more information about common Identity Vault and driver set packages, see "Creating Identity Vault and Driver Set Packages" on page 226.

16e Click **OK**.

17 Drag and drop a development driver from the palette. Follow the steps in "Creating a Development Driver" on page 195 in case you wish to install a development driver.

18 In the Modeler, right-click the development driver, then click **Driver > Properties**.

19 In the Properties window, click **Packages** to install the base package on the driver.

NOTE: Ensure that the package created in Step 7 is listed.

---

20 The package list is initially filtered by driver types. To see all available driver packages, deselect **Show only applicable package versions**.

21 Click **OK**.

Configuring Initial Settings

After you create your custom base package, configure the initial package settings you want to use for the driver. When you install the driver, the driver's initial settings create a set of objects that the driver needs for startup. The initial settings for a driver are specified as ds-object code. The ds-object code installs driver shim parameters, driver start options, named passwords, GCVs, and filters.

When you create a package, the initial settings XML for the package is empty by default. This is displayed in the Package Properties window. Unless you are extremely proficient with XML and possess a good understanding of OpenText Identity Manager schema, we recommend that you populate your initial settings from an existing template.

You can use a working driver as a template, if you want your package to use specific settings from that driver. For example, if you want to create a custom OpenText eDirectory package, you can use an OpenText eDirectory driver as your development driver and populate your initial settings from the development driver.

If you only want to include minimum initial settings in your package and configure them manually, you can also add an empty Generic App driver. You can only add certain driver properties as ds-attribute objects in the initial settings, as listed in the table below.

ds-attribute Object	Identity Vault Mapped Driver Attribute	Type	Description
`name`	`CN`	String	Specifies the name of the driver.
`application-schema`	`DirXML-ApplicationSchema`	XML	Specifies the schema of the application to which the driver connects. Each application has its own schema, but OpenText Identity Manager does not necessarily use all classes or attributes from an application schema. For more information about application schema, see "Managing a Copy of an Application Schema" on page 166

ds-attribute Object	Identity Vault Mapped Driver Attribute	Type	Description
configuration-manifest	DirXML-ConfigManifest	XML	Contains the driver health configuration settings for the driver. These settings allow you to monitor the state of the driver and configure the driver to perform actions automatically depending on the driver's health state. For more information about the Driver Health Configuration, see "Driver Health Configuration" on page 93.
driver-filter-xml	DirXML-DriverFilter	XML Can include a class and attributes.	Specifies how the driver should filter incoming data. We recommend that you do not use this attribute to configure the base driver filter, instead create filter extension objects. For more information about creating filter extension objects, see "Adding Filter Extensions" on page 234
reciprocal-links	DirXML-ReciprocalAttrMap	XML	The Reciprocal Attributes property page lets you create and manage backlinks between objects. For example, the Group object includes a Members attribute that contains pointers to all User objects that belong to that group. Similarly, each User object includes a Group Membership attribute that points to the Group objects of which that user is a member. These two-way links between objects are known as reciprocal mappings. For more information about reciprocal links, see "Reciprocal Attributes" on page 104
driver-image	DirXML-DriverImage	String	Driver image is a 64-bit encoded image that represents the driver in the iManager Web interface.

ds-attribute Object	Identity Vault Mapped Driver Attribute	Type	Description
trace-name	DirXML-TraceName	String	Trace name helps you track trace messages. This name appears in the driver trace messages. Use a trace name if the driver name is very long. For more information about Driver Trace, see “Driver Trace Levels” on page 107
trace-file	DirXML-TraceFile	String	When you set a value to this field, all Java information for the driver is written to the file. The value for this field is the path for that file. As long as the file is specified, Java information is written to this file. If you do not need to debug Java, leave this field blank.
trace-file-encoding	DirXML-TraceFileEncoding	String	Specifies the trace file encoding. The trace file uses the system’s default encoding. You can specify another encoding if desired.
trace-level	DirXML-TraceLevel	Integer	You can add a trace to your driver. With the driver trace level set, DS Trace displays driver-related OpenText Identity Manager events, at the level of detail specified by the driver trace level as the engine processes the events. The driver trace level affects only the driver or driver set where it is set.
trace-size-limit	DirXML-TraceSizeLimit	Integer	Allows you to set a limit for the Java trace file. If you set the file size to Unlimited, the file grows in size until there is no disk space left.
java-module	DirXML-JavaModule	String	Specifies the driver shim XML configuration the driver uses. For example, com.novell.nds.dirxml.driver.nds.DriverShimImpl or com.novell.idm.driver.ComposerDriverShim.

ds-attribute Object	Identity Vault Mapped Driver Attribute	Type	Description
native-module	DirXML-NativeModule	String	Specifies the name of the DLL file that will be instantiated for the application shim component of the driver.
driver-trace-level	DirXML-DriverTraceLevel	Integer	With the trace set, DS Trace displays OpenText Identity Manager and DirXML events as the engine processes the events. The trace level affects each driver in the driver set. Use the trace level for troubleshooting issues with the drivers when they are deployed. DS Trace displays the output of the specified trace level.
log-limit	DirXML-LogLimit	Integer	Allows you to set a limit for the log file.
shim-auth-id	DirXML-ShimAuthID	String	Specifies the application user ID. This ID is used to pass Identity Vault subscription information to the application. If you enabled SSL/TLS for OpenText eDirectory drivers, this option is greyed out. For more information about shim, see "Custom Shims" on page 511
shim-auth-server	DirXML-ShimAuthServer	String	The server that the driver is associated with.
log-events	DirXML-LogEvents	XML	Specifies the types of events you want the driver to log in the audit log. For example, you can configure the driver to log errors, warning, or specific events like object modifications. By default, the driver uses the settings from the driver set, as specified in the Log Level tab in the driver set Properties window. For more information about configuring log levels, see "Driver Set Log Levels" on page 79.

ds-attribute Object	Identity Vault Mapped Driver Attribute	Type	Description
shim-config-info	DirXML-ShimConfigInfo	XML	Specifies the Driver Parameters settings displayed in the Properties window for the driver.
global-config-values	DirXML-ConfigValues	XML	Specifies any GCVs configured on the driver. For more information about GCVs, see “Driver Global Configuration Values” on page 91.
global-engine-values	DirXML-EngineControlValues	XML	Specifies the engine control values used by all drivers, including the Subscriber channel retry interval and maximum OpenText eDirectory replication wait time. For more information about engine control values, see “Engine Control Values” on page 87.
driver-start-option	DirXML-DriverStartOption	Integer	Specifies the default startup option for the driver. For more information about driver startup options, see “Startup Option” on page 86.
named-password	DirXML-NamedPasswords	String	Specifies any named passwords configured on the driver. For more information about named passwords, see “Driver Named Passwords” on page 102.
driver-cache-limit	DirXML-DriverCacheLimit	Integer	Specifies the limit to the driver cache file. By default, the driver cache (file) size is limited only by available disk space. This is the recommended setting. For more information about driver-cache-limit, see“Driver Configuration” on page 85
driver-password	DirXML-ShimAuthPassword	String	The driver shim password is prompted for during import.

Complete the following steps to add initial settings to your base package.

1. In the Outline view, right-click the base package, then select **Properties**.
2. Click **Initial Settings**.

3 Click **Populate From Template**.

WARNING: When you populate your initial settings from a template, Designer overwrites any XML currently in the Initial Settings window. If you have any previously-customized XML, ensure that you save the existing XML before clicking **Populate From Template**.

4 In the Model Browser window, select the driver you want to use as a template, then click **OK**.

NOTE: You can use any driver currently available in your workspace to populate your Initial Settings window.

5 Modify the package initial settings as necessary for your environment.

6 When finished, click **OK**.

Working with Package Prompts

After you create a base package, we recommend you create package prompts for use in your packages. Package prompts should be stored in the base package, rather than in specific feature sub-packages, so that all feature packages can use the configured prompts if needed.

Understanding Package Prompts

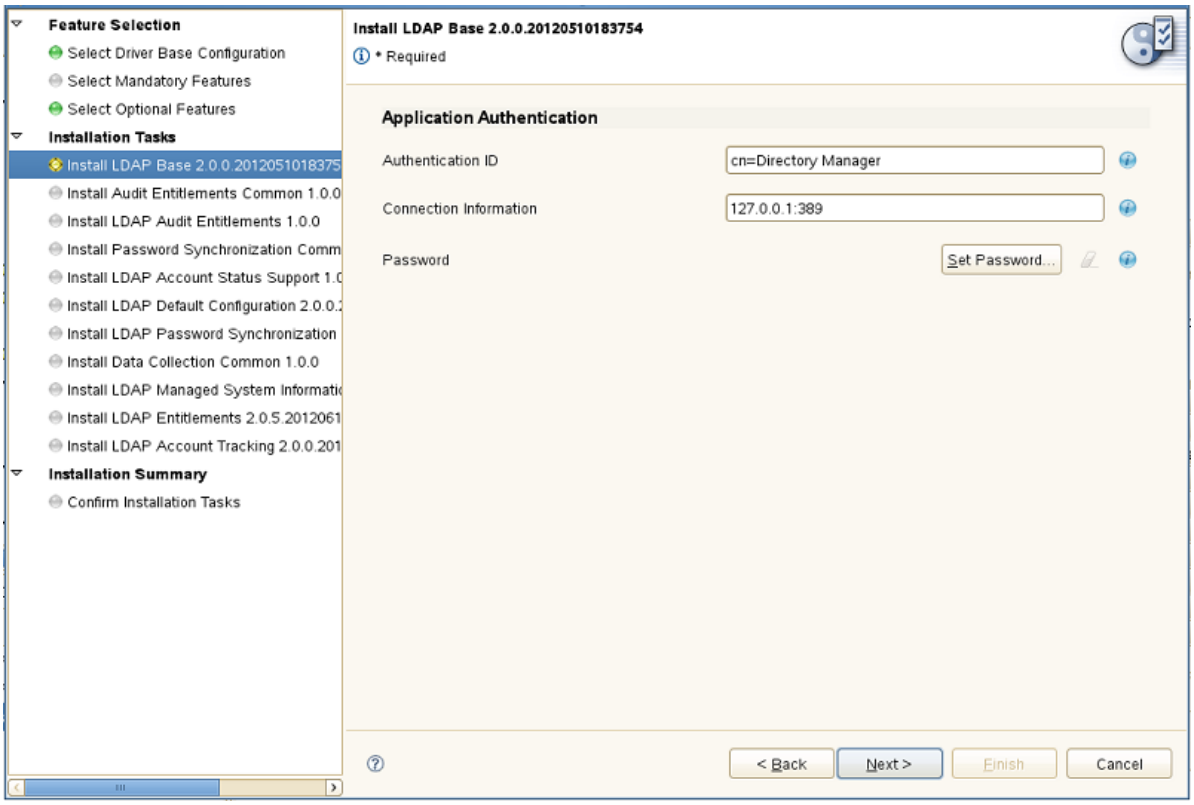
Package prompts allow users to configure the packages included in a driver during the driver installation process. When a user installs a driver, they provide configuration information necessary for that user's environment.

Some packages include default configuration information built into the package by the package developer, but many configuration properties must be specified at the time of installation. For example, users may need to specify the IP address of the target system or the name of the Identity Vault container used to store user or group information.

The Driver Configuration Wizard provides one or more windows that includes fields where the user can configure the driver. The windows the Driver Configuration Wizard displays are package prompts. You can use package prompts to modify any of the properties of a driver, including the driver name, driver configuration parameters, GCVs, or job parameters.

Prompts are stored as Resource objects and are typically stored in the base package of a driver. Each prompt Resource object can contain one or more fields, which is displayed to the user in the Driver Configuration Wizard. Each prompt corresponds to a window within the Wizard and can be required or optional, as necessary.

The following graphic provides an example of a default Initial Settings prompt:



NOTE: ◆A package can contain no prompts or many prompts, depending on the needs of the driver.

- When you install a package, the Driver Configuration Wizard displays package prompts according to the Order parameter value of each prompt. To configure the order in which your prompts appear, right-click the prompt resource in the Outline view and select **Properties**, specify the value you want to use for the **Order** parameter, and click **OK**.
- Each package prompt is a Resource object of the type `application/vnd.novell.dirxml.pkg-prompt+xml`.

Designer creates a default pair of XSL style sheets when you create a new package prompt. You can modify those style sheets to fit your needs. Designer uses XSL style sheets to transform both the prompt fields displayed in the Driver Configuration Wizard and package items contained in the target packages specified for the prompt.

The **prompt transform** configures the way the prompt looks in the Wizard, while the **target transform** takes information users input using the prompts and modifies objects in your environment depending on that input. Prompts can set values in GCVs and be used to configure specific features of the driver, such as using entitlements or synchronizing passwords. For more information about package transformations, see "Understanding Package Prompt Transformations" on page 208.

Understanding Package Prompt Types

There are eight types of default package prompts available in Designer:

- Driver Name

- Global Configuration
- Initial Settings
- Job
- Remote Loader
- Upgrade Settings
- MSysInfo Classification
- Custom

Each type of package prompt has its own set of default fields. However, you can add new fields to a package prompt to configure other driver configuration properties, as necessary in your environment. When you add a new prompt field, Designer creates a GCV for that field.

The following sections describe the different default package prompt types.

NOTE: ◆You can only generate package prompt resources of the Driver Name, Initial Settings, Remote Loader, or Upgrade Settings types from the Package Catalog.

- To generate Global Configuration and Job package prompts, you must first create a corresponding object, then generate a prompt for the object and add the prompt to a package.
- To generate a Custom package prompt, you must create a Resource object of the `application/vnd.novell.dirxml.pkg+prompt+xml` type.
- MSysInfo Classification package prompts are created outside of the package prompt interface.
- You can only generate Driver Name and Remote Loader package prompt resources on a base package.

Driver Name

This type of package prompt allows users to specify the name of the driver. The only prompt field included in this package prompt is **Driver Name**.

Field Display Name	Field Attribute Name
Driver Name	`name`

This package prompt is only available for base packages.

Global Configuration

This type of package prompt allows users to modify the properties of one or more GCV resources.

For more information about creating a Global Configuration package prompt, see “Creating Global Configuration Prompts” on page 223.

Initial Settings

This type of package prompt allows users to configure the initial driver configuration properties of a driver object. For example, a user can specify the connection information and password they want to use for the driver.

For more information about configuring the initial settings for a driver, see "Configuring Initial Settings" on page 199.

Field Display Name	Field Attribute Name
Authentication ID	`shim-auth-id`
Connection Information	`shim-auth-server`
Password	`shim-auth-password`

Job

This type of package prompt allows users to modify specific parameters of a job contained in the package.

Remote Loader

This type of package prompt allows users to configure the Remote Loader settings for the driver. If your driver supports the Remote Loader, you must include the Remote Loader package prompt in your package. Packages typically display the Remote Loader package prompt last during driver installation.

Field Display Name	Field Attribute Name
Connect To Remote Loader	`use-remote-loader`
Host Name	`rl-hostname`
Port	`rl-port`
KMO	`rl-kmo`
Other parameters	`rl-other`
Remote Password	`rl-password`
Driver Password	`driver-password`
Manager Password	`ManagerPassword`

This package prompt is only available for base packages.

Upgrade Settings

This type of package prompt contains style sheets that maintain your custom package settings so that they are not overwritten when you upgrade or downgrade the package.

The Upgrade Settings package prompt contains no prompt fields.

MSysInfo Classification

This type of package prompt allows users to specify the classification of a particular managed system and the type of environment the managed system provides. The Reporting module can then classify the driver by managed system or environment in reports.

NOTE: This package prompt is typically only used in specialized drivers like OpenText eDirectory.

Users can select one of the following options for the classification of a managed system:

- Mission-Critical
- Vital
- Not-Critical
- Other

Users can select one of the following options for the environment of a managed system:

- Development
- Test
- Staging
- Production
- Other

Custom

This type of package prompt can be customized to modify anything the package installs. The target of a custom package prompt is any object in the package that you want users to be able to change when installing and configuring the driver. For example, if you want users to modify a policy during the installation process, you can create a custom package prompt and specify the policy as the target for the prompt.

For more information about creating custom package prompts, see "Creating Package Prompt Resources" on page 224.

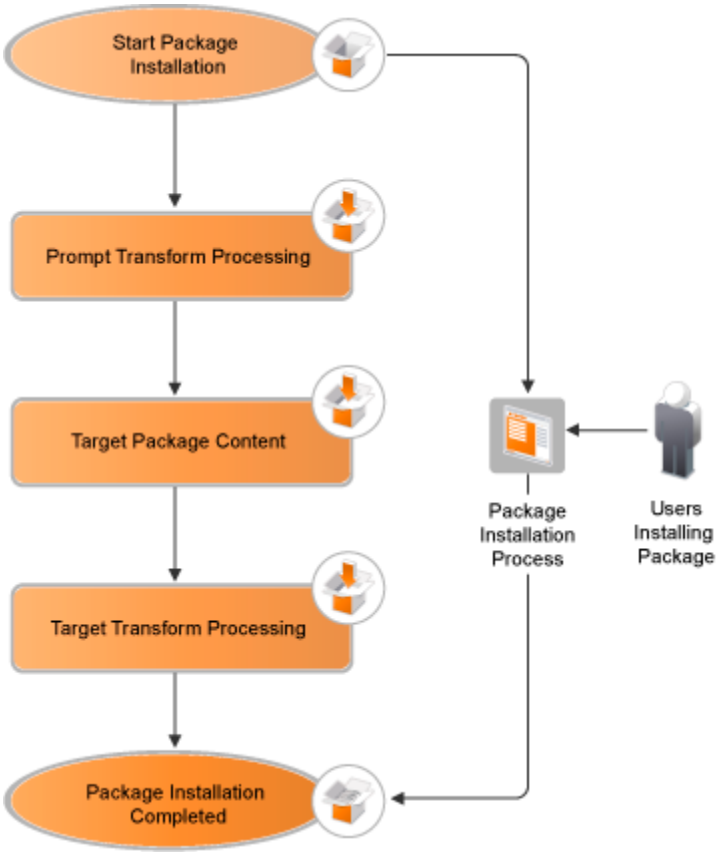
Understanding Package Prompt Transformations

When you install a package, Designer performs the following tasks for each prompt that belongs to the package:

- Reads the prompt
- Applies the prompt transform XSL on the prompt XML

- Displays the transformed prompt in the Driver Configuration Wizard
- Receives the values specified by the user in the Driver Configuration Wizard
- Applies the target transform XSL on the target object using the values specified by the user and the initial package settings

The following diagram displays the Designer workflow for prompt transformations:



Prompt transforms are typically used for **conditional prompting**, where the Driver Configuration Wizard only displays a prompt if specific conditions are met. For example, when you install a driver, the Driver Name prompt allows you to specify a name for the driver. However, when you run view the driver properties after installation, Designer does not display the Driver Name prompt.

Target transforms are typically used to modify different types of targets during the driver installation process. For example, target transforms allow you to modify the named password used by a particular driver, based on the password the user specifies in a package prompt.

NOTE: Most package developers can use an existing XSL style sheet for their package-creation needs. However, advanced users may need to customize the XSL style sheets. To customize prompt and target transforms, you should understand the style sheets and the inputs the style sheets receive. See the sections below for information about default style sheets and inputs.

Each transform includes three XML documents, **defsDoc**, **curDoc**, and **npDoc**, as well as the boolean **propertyWizard** flag.

These four components allow you to apply a transform to a prompt or target, depending on your needs. You add defsDoc, curDoc, npDoc, or propertyWizard to your transform as parameters in the XSL code. For more information about the transform parameters, see the following sections.

defsDoc

This XML parameter contains the prompts, or configuration value definitions, including the values specified by the user on the prompt page.

Sample document:

```
    <configuration-values>
        <definitions>
            <header display-name="Authentication"/>
            <definition display-name="SAP User ID" mandatory="true"
name="shim-auth-id" type="string">
              <description>The ID of the User this driver will use for SAP
Logon.  This is referred to as 'User' in the SAP Logon screen.</
description>
               <value>idmdriver</value>
            </definition>
            <definition display-name="SAP User Password" mandatory="true"
name="shim-auth-password" type="password-ref">
              <description>The User password this driver will use for SAP
Logon.  This is referred to as 'Password' in the SAP Logon screen.</
description>
               <value>shim-auth-password</value>
            </definition>
        </definitions>
    </configuration-values>
```

curDoc

In the case of an upgrade or downgrade using the Installation Wizard, this parameter contains the XML content of the currently installed prompt target. In the case of an initial install using the Driver Configuration Wizard, this document is empty.

Sample document (only an excerpt, as these docs are rather large):

```
    <ds-attributes>
        <ds-attribute ds-attr-name="shim-auth-id">
            <ds-value>idmdriver</ds-value>
        </ds-attribute>
        <ds-attribute ds-attr-name="shim-auth-server">
            <ds-value>127.0.0.1</ds-value>
        </ds-attribute>
        <ds-attribute ds-attr-name="driver-start-option">
            <ds-value>2</ds-value>
        </ds-attribute>
    </ds-attributes>
```

npDoc

In the case of an upgrade or downgrade, this parameter contains an XML representation of all named passwords available on the prompt target.

Only the names of existing passwords are available, not their values. If a named password has been set using a prompt, both its name and value are available.

To set a named password, append the following structure to the transform target:

```
<ds-attribute ds-attr-name="named-password">
    <ds-value display-name="Password 1" name="pwd1">1</ds-value>
    <ds-value display-name="Password 2" name="pwd2">2</ds-value>
</ds-attribute>
```

NOTE: ◆The transform target must support named passwords.

- You cannot get or modify passwords using a handle to the npDoc document. For security reasons, the value of the password itself is never displayed.

Sample document:

```
<named-passwords>
    <named-password name="promptedPwd">promptedValue</named-password>
    <named-password name="existingPwd"/>
</named-passwords>
```

propertyWizard Flag

This boolean parameter indicates if the package is installed from the Installation Wizard, which is launched from the package Properties window, or from the Driver Configuration Wizard, which Designer launches when you install a new driver. The possible options are `true` (Installation Wizard) or `false` (Driver Configuration Wizard).

This parameter allows you to configure a package prompt to be displayed or hidden depending on the wizard. For the Driver Name prompt, this parameter is set to `false` by default, so that Designer only prompts users for the driver name in the Driver Configuration Wizard.

Example Default Prompt Transformations

As discussed previously, each of the default package prompt types contains both a prompt transformation and a target transformation. The following subsections provide examples of some of the default prompt transformation stylesheets.

Driver Name

The default prompt transformation for a Driver Name package prompt uses the propertyWizard flag to check if the user is viewing the prompt in the Installation Wizard or Driver Configuration Wizard, then pre-populates the prompt with an existing value, if a driver name already exists.

```
  <xsl:param name="propertyWizard"/>
  <xsl:template match="header[@driver-name='true']">
    <xsl:if test="$propertyWizard='false'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
  <xsl:template match="definition[@driver-name='true']">
    <xsl:if test="$propertyWizard='false'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
  <!-- pre-populate prompts with existing values -->
  <xsl:template match="definition/value">
    <xsl:variable name="name" select="../@name"/>
    <xsl:variable name="curVal">
      <xsl:choose>
        <xsl:when test="$curDoc//ds-value[../@ds-attr-name=$name]/text()">
          <xsl:value-of select="$curDoc//ds-value[../@ds-attr-name=$name]/
text()"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$curDoc//value[../@name=$name]/text()"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:choose>
      <!-- backfilling from current value -->
      <xsl:when test="$curVal">
        <value>
          <xsl:value-of select="$curVal"/>
        </value>
      </xsl:when>
      <!-- no current value found -->
      <xsl:otherwise>
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <!-- identity transformation template -->
  <xsl:template match="node()|@*">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
```

Initial Settings

The default prompt transformation for an Initial Settings package prompt pre-populates the prompt fields with existing values, if applicable.

```
  <xsl:param name="propertyWizard"/>
  <xsl:template match="header[@driver-name='true']">
    <xsl:if test="$propertyWizard='false'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
  <xsl:template match="definition[@driver-name='true']">
    <xsl:if test="$propertyWizard='false'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
  <!-- pre-populate prompts with existing values -->
  <xsl:template match="definition/value">
    <xsl:variable name="name" select="../@name"/>
    <xsl:variable name="curVal">
      <xsl:choose>
        <xsl:when test="$curDoc//ds-value[../@ds-attr-name=$name]/text()">
          <xsl:value-of select="$curDoc//ds-value[../@ds-attr-name=$name]/
text()"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$curDoc//value[../@name=$name]/text()"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:choose>
      <!-- backfilling from current value -->
      <xsl:when test="$curVal">
        <xsl:variable name="checkRemote">
          <xsl:choose>
            <xsl:when test="$name='shim-auth-server' or $name='shim-auth-
password'">
              <xsl:value-of select="'true'"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="'false'"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
        <xsl:choose>
          <xsl:when test="$checkRemote='true' and starts-with($curVal,
'REMOTE')">
            <value>
              <xsl:value-of select="substring-after($curVal, ')')"/>
            </value>
          </xsl:when>
```

```
          <xsl:otherwise>
            <value>
              <xsl:value-of select="$curVal"/>
            </value>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <!-- no current value found -->
      <xsl:otherwise>
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <!-- identity transformation template -->
  <xsl:template match="node()|@*">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
```

Example Default Target Transformations

The following subsections provide examples of some of the default target transformation stylesheets.

Global Configuration

The default target transformation for a Global Configuration package prompt applies the specified prompt values to a global configuration object.

```
  <xsl:param name="propertyWizard"/>
  <!-- handle non-existing named passwords -->
  <xsl:template match="ds-attributes">
    <xsl:copy>
      <xsl:apply-templates select="@*"/>
      <xsl:choose>
        <!-- no named passwords defined in initial settings -->
        <xsl:when test="count(ds-attribute[@ds-attr-name='named-
password'])=0">
          <ds-attribute ds-attr-name="named-password">
            <xsl:for-each select="$npDoc//named-passwords/named-
password[count($defsDoc//definition[@type='password-ref']/
value[text()=@name])>0]">
              <ds-value display-name="a" name="a">bb</ds-value>
            </xsl:for-each>
          </ds-attribute>
        </xsl:when>
        <!-- named passwords defined in initial settings -->
        <xsl:otherwise>
          <xsl:apply-templates select="node()"/>
        </xsl:otherwise>
      </xsl:choose>
```

```
    </xsl:copy>
  </xsl:template>
  <!-- handle existing named passwords -->
  <xsl:template match="ds-attribute[@ds-attr-name='named-password']">
    <xsl:copy>
      <xsl:apply-templates select="@*"/>
      <xsl:for-each select="ds-value">
        <xsl:copy>
          <xsl:apply-templates select="@*"/>
          <xsl:variable name="npName" select="@name"/>
          <xsl:variable name="npValue" select="$npDoc//named-passwords/
named-password[@name=$npName]/text()"/>
          <xsl:choose>
            <xsl:when test="string-length($npValue)>0">
              <xsl:value-of select="$npValue"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="."/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:copy>
      </xsl:for-each>
    </xsl:copy>
  </xsl:template>
  <!-- inject prompt values into target definitions -->
  <xsl:template match="definition/value">
    <xsl:variable name="name" select="../@name"/>
    <xsl:variable name="promptVal" select="$defsDoc//value[../
@name=$name]"/>
    <xsl:choose>
      <!-- inject value from prompt -->
      <xsl:when test="$promptVal">
        <xsl:copy>
          <xsl:value-of select="$promptVal"/>
        </xsl:copy>
      </xsl:when>
      <!-- no current value found -->
      <xsl:otherwise>
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <!-- identity transformation template -->
  <xsl:template match="node()|@*">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
```

Remote Loader

The default target transformation for a Remote Loader package prompt handles Remote Loader-specific prompt fields. The target transformation also provides the Remote Loader parameters and password to the Initial Settings package prompt to use in the Connection Information and Password fields.

```
<xsl:param name="propertyWizard"/>
  <xsl:template match="ds-attribute[@ds-attr-name='driver-password']"/>
  <!-- Remove the native module if we are running remote -->
  <xsl:template match="ds-attribute[@ds-attr-name='native-module']">
    <xsl:variable name="useRemoteLoader" select="$defsDoc//
definition[@name='use-remote-loader']/value/text()"/>
    <xsl:if test="$useRemoteLoader='false'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
  <!-- Replace the java module with the remote shim if we are running
remote -->
  <xsl:template match="ds-attribute[@ds-attr-name='java-module']/ds-value/
text()">
    <xsl:variable name="useRemoteLoader" select="$defsDoc//
definition[@name='use-remote-loader']/value/text()"/>
    <xsl:choose>
      <xsl:when test="$useRemoteLoader='true'">
        <xsl:value-of
select="'com.novell.nds.dirxml.remote.driver.DriverShimImpl'"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="ds-attributes">
    <xsl:variable name="useRemoteLoader" select="$defsDoc//
definition[@name='use-remote-loader']/value/text()"/>
    <xsl:variable name="rlHost" select="$defsDoc//definition[@name='rl-
hostname']/value/text()"/>
    <xsl:variable name="rlPort" select="$defsDoc//definition[@name='rl-
port']/value/text()"/>
    <xsl:variable name="rlKMOTemp" select="$defsDoc//definition[@name='rl-
kmo']/value/text()"/>
    <xsl:variable name="rlKMO">
      <xsl:choose>
        <xsl:when test="string-length($rlKMOTemp)>0">
          <xsl:choose>
            <xsl:when test="contains($rlKMOTemp, ' ')">
              <xsl:variable name="c1" select="concat(&quot;&apos;&quot;,
$rlKMOTemp)"/>
              <xsl:variable name="c2" select="concat($c1,
&quot;&apos;&quot;)"/>
              <xsl:value-of select="concat(' kmo=', $c2)"/>
            </xsl:when>
```

```
            <xsl:otherwise>
              <xsl:value-of select="concat(' kmo=', $rlKMOTemp)"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="''"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="rlOtherTemp" select="$defsDoc//
definition[@name='rl-other']/value/text()"/>
    <xsl:variable name="rlOther">
      <xsl:choose>
        <xsl:when test="string-length($rlOtherTemp)>0">
          <xsl:value-of select="concat(' ', $rlOtherTemp)"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="''"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="rlPwd" select="$npDoc//named-password[@name='rl-
password']/text()"/>
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
      <xsl:if test="$useRemoteLoader='true'">
        <!-- inject the driver password if running remote -->
        <xsl:for-each select="$npDoc//named-passwords/named-
password[@name='driver-password']/text()">
          <ds-attribute ds-attr-name="driver-password">
            <ds-value>
              <xsl:value-of select="."/>
            </ds-value>
          </ds-attribute>
        </xsl:for-each>
        <!-- Add a java module attribute node if one does not exist -->
        <xsl:choose>
          <xsl:when test="ds-attribute[@ds-attr-name='java-module']">
            <!--  Do nothing -->
          </xsl:when>
          <xsl:otherwise>
            <ds-attribute ds-attr-name="java-module">
              <ds-
value>com.novell.nds.dirxml.remote.driver.DriverShimImpl</ds-value>
            </ds-attribute>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:if test="$rlHost">
          <!-- Add a shim-auth-server attribute node if one does not exist
-->
          <xsl:choose>
            <xsl:when test="ds-attribute[@ds-attr-name='shim-auth-server']/
ds-value/text()">
              <!--  Do nothing -->
```

```
            </xsl:when>
            <xsl:otherwise>
              <ds-attribute ds-attr-name="shim-auth-server">
               <ds-value>REMOTE(hostname=<xsl:value-of select="$rlHost"/>
port=<xsl:value-of select="$rlPort"/>
                  <xsl:value-of select="$rlKMO"/>
                  <xsl:value-of select="$rlOther"/>)</ds-value>
              </ds-attribute>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:if>
        <xsl:if test="$rlPwd">
         <!-- Add a shim-auth-password attribute node if one does not exist
-->
          <xsl:choose>
            <xsl:when test="ds-attribute[@ds-attr-name='shim-auth-
password']/ds-value/text()">
              <!--  Do nothing -->
            </xsl:when>
            <xsl:otherwise>
              <ds-attribute ds-attr-name="shim-auth-password">
               <ds-value>REMOTE(<xsl:value-of select="$rlPwd"/>)</ds-value>
              </ds-attribute>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:if>
      </xsl:if>
    </xsl:copy>
  </xsl:template>
  <!-- Fix up shim-auth-server if running remote and one already exists -->
  <xsl:template match="ds-attribute[@ds-attr-name='shim-auth-server']/ds-
value/text()">
    <xsl:variable name="useRemoteLoader" select="$defsDoc//
definition[@name='use-remote-loader']/value/text()"/>
    <xsl:variable name="rlHost" select="$defsDoc//definition[@name='rl-
hostname']/value/text()"/>
    <xsl:variable name="rlPort" select="$defsDoc//definition[@name='rl-
port']/value/text()"/>
    <xsl:variable name="rlKMOTemp" select="$defsDoc//definition[@name='rl-
kmo']/value/text()"/>
    <xsl:variable name="rlKMO">
      <xsl:choose>
        <xsl:when test="string-length($rlKMOTemp)>0">
          <xsl:choose>
            <xsl:when test="contains($rlKMOTemp, ' ')">
              <xsl:variable name="c1" select="concat(&quot;&apos;&quot;,
$rlKMOTemp)"/>
              <xsl:variable name="c2" select="concat($c1,
&quot;&apos;&quot;)"/>
              <xsl:value-of select="concat(' kmo=', $c2)"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="concat(' kmo=', $rlKMOTemp)"/>
            </xsl:otherwise>
          </xsl:choose>
```

```
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="''"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="rlOtherTemp" select="$defsDoc//
definition[@name='rl-other']/value/text()"/>
    <xsl:variable name="rlOther">
      <xsl:choose>
        <xsl:when test="string-length($rlOtherTemp)>0">
          <xsl:value-of select="concat(' ', $rlOtherTemp)"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="''"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="$useRemoteLoader='true'">
        <xsl:variable name="curVal" select="."/>
        <xsl:variable name="tmpVal"
select="concat(concat('REMOTE(hostname=', $rlHost), ' port=')"/>
        <xsl:variable name="remoteVal" select="concat(concat($tmpVal,
$rlPort), $rlKMO)"/>
       <xsl:variable name="withKMO" select="concat($remoteVal, $rlOther)"/
>
        <xsl:variable name="withOther" select="concat($withKMO, ')')"/>
        <xsl:variable name="serverVal" select="concat($withOther,
$curVal)"/>
        <xsl:value-of select="$serverVal"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <!-- Fix up shim-auth-password if running remote and one already exists -
->
  <xsl:template match="ds-attribute[@ds-attr-name='shim-auth-password']/
ds-value/text()">
    <xsl:variable name="useRemoteLoader" select="$defsDoc//
definition[@name='use-remote-loader']/value/text()"/>
    <xsl:variable name="rlPwd" select="$npDoc//named-password[@name='rl-
password']/text()"/>
    <xsl:choose>
      <xsl:when test="$useRemoteLoader='true'">
        <xsl:variable name="curVal" select="."/>
        <xsl:variable name="remoteVal" select="concat(concat('REMOTE(',
$rlPwd), ')')"/>
```

```
        <xsl:variable name="pwdVal" select="concat($remoteVal, $curVal)"/>
        <xsl:value-of select="$pwdVal"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <!-- identity transformation template -->
  <xsl:template match="node()|@*">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
```

Examples of Modified Prompt Transformations

In this section, we provide a few examples to demonstrate how you can modify a prompt transform and why modifying a prompt transform can be useful.

Use Case 1: Need to configure different behavior for package installation through the Driver Configuration Wizard and the Installation Wizard

If you upgrade a package using the Installation Wizard, Designer does not need to prompt you for the driver name, as the driver name should already be configured.

To avoid the Wizard prompting you for the driver name, use the flag `propertyWizard` in the prompt transform. Depending on the flag, we remove the given prompt from prompt display.

Sample XSL code:

```
<xsl:template match="header[@driver-name='true']">
          <xsl:if test="$propertyWizard='false'">
                <xsl:copy>
                        <xsl:apply-templates select="@*|node()"/>
                </xsl:copy>
          </xsl:if>
</xsl:template>
```

Use Case 2: Need to pre-fill prompts with existing values during the upgrade process

During an upgrade or downgrade, Designer ensures that Designer displays the values you entered during the initial installation. The user therefore does not need to remember all the values specified during the first installation.

For each definition in the input document (in this case, the prompt document), Designer tries to find the corresponding definition in the current document (curDoc). When Designer finds a matching definition, the application stores the corresponding value in a temporary variable, `curVal`.

Designer then populates the prompt document with the `curVal` value and displays the pre-filled prompts to the user during the upgrade or downgrade process.

Sample XSL code:

```
<xsl:template match="definition/value">
    <xsl:variable name="name" select="../@name"/>
    <xsl:variable name="curVal">
      <xsl:choose>
        <xsl:when test="$curDoc//ds-value[../@ds-attr-name=$name]/text()">
          <xsl:value-of select="$curDoc//ds-value[../@ds-attr-name=$name]/
text()"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of
select="$curDoc//value[../@name=$name]/text()"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:choose>
      <!-- backfilling from current value -->
      <xsl:when test="$curVal">
        <value>
          <xsl:value-of select="$curVal"/>
        </value>
      </xsl:when>
      <!-- no current value found -->
      <xsl:otherwise>
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

Example of Modified Target Transformation

In this section, we provide an example to demonstrate how you can modify a target transform and why modifying a target transform can be useful.

Use Case: Need to provide the driver name at the necessary place during target transformation

In this case, you want to add the driver name to the initial data so that the driver name prompt changes the name in all necessary locations on the driver.

Sample XSL code:

```
<xsl:template match="ds-attributes">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
      <xsl:if test="$propertyWizard='false' and boolean(ds-attribute[@ds-
attr-name='name']/ds-value)=false()">
        <!-- Make sure we have a name when called from the DCW -->
        <xsl:variable name="promptVal" select="$defsDoc//value[../
@name='name']"/>
        <xsl:variable name="driverName">
          <xsl:choose>
            <!-- use prompt value -->
            <xsl:when test="$promptVal">
              <xsl:value-of select="$defsDoc//value[../@name='name']/
text()"/>
            </xsl:when>
            <!-- no prompt value found, use default value -->
            <xsl:otherwise>Driver</xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
        <ds-attribute ds-attr-name="name">
          <ds-value>
            <xsl:value-of select="$driverName"/>
          </ds-value>
        </ds-attribute>
      </xsl:if>
    </xsl:copy>
</xsl:template>
```

Adding Default Package Prompts

To add package prompts to a base package:

1. Verify that you have created a base package. Otherwise, follow "Creating a Base Package" on page 197 to create a new base package.
2. Right-click the package in the package catalog and select **Generate Prompt Resource**.
3. Select the type of package prompt you want to configure.

   NOTE: You can only create one of each type of prompt for a particular package.

4. In the package catalog, expand the package version and Resources directory.
5. Right-click the new package prompt and select **Properties**.
6. Verify the type of package prompt.
7. Specify the order in which you want the Driver Configuration Wizard to display the current package prompt. The Wizard displays prompts in ascending order starting from `0`.
8. Verify the target displayed is correct for the package prompt. If you want to add the prompt to a different package, click **Add**, browse to the package, and click **OK**.
9. Click the Prompts tab. The Properties window displays what the current package prompt looks like in the Driver Configuration Wizard.

You now have default package prompts created and you can edit and change these prompts for your own needs.

Creating Custom Package Prompts

In addition to adding default, auto-generated prompts to your packages, you can create a custom prompt to modify a specific GCV object in your package or create a package prompt resource to modify any non-GCV target object in your package.

Creating Global Configuration Prompts

You can create a package prompt that modifies a GCV object contained in your custom package.

NOTE: To create a Global Configuration prompt, you must first install the base package on the development driver.

For more information, see the "Global Configuration Value Definition Editor" in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

To create and configure a Global Configuration package prompt, complete the following steps:

1. Install the base package you want to use on your development driver. For more information about installing the development driver, see "Creating a Development Driver" on page 195.
2. In the Outline view, right-click the driver name and select **New > Global Configuration**.
3. Specify a name for the new GCV resource object and click **OK**.
4. In the Outline view, right-click the new GCV resource object and select **Add to Package**.
5. Select the base package where you want to add the GCV resource and click **OK**.
6. In the Outline view, navigate to the base package and expand **Global Configurations**.
7. Right-click the new GCV resource object and select **Generate Prompt Resource**. Designer creates a new package prompt for the GCV in the Resources directory.
8. Right-click the new GCV package prompt and select **Properties**.
9. Verify that the target of the package prompt is the GCV resource object you created.
10. Specify the order in which you want the Driver Configuration Wizard to display the GCV package prompt. The Wizard displays prompts in ascending order starting from `0`.
11. Click **Prompts**.
12. Click **Add** to add each new prompt you want to include in the GCV package prompt Resource object. For information about adding new prompts, see "Adding Prompts" on page 225.
13. When finished adding prompts, click **Apply**.
14. Click **Prompt Transformation**. This window allows you to configure how you want to display the prompt in the Driver Configuration Wizard.
15. Modify the default Global Configuration transform as necessary for your GCV package prompt. For more information about the default Global Configuration prompt transform, see "Global Configuration" on page 206. For more information about prompt transforms, see "Understanding Package Prompt Transformations" on page 208.
16. Click **Apply**.
17. Click **Target Transformation**. This window allows you to configure how you want to modify the target of the transform.

18 Modify the default Global Configuration transform as necessary for your GCV package prompt. For more information about the default Global Configuration target transform, see "Global Configuration" on page 206. For more information about target transforms, see "Understanding Package Prompt Transformations" on page 208.

19 Click **OK**.

Creating Package Prompt Resources

You can create custom package prompts directly as resource objects themselves. You can create a package prompt to modify any object the package installs on the driver.

To create a custom package prompt, complete the following steps:

1 In the Outline view, right-click the development driver and select **New > Resource**.

2 Specify the name you want to use for the custom prompt.

3 In the Content type drop-down menu, select **application/vnd.novell.dirxml.pkg+prompt+xml**.

4 Clear **Open the editor after creating the object** and click **OK**.

5 (Optional) If prompted to save, click **Yes**.

6 In the Outline view, right-click the custom package prompt Resource object and select **Add to Package**.

7 Select the base package where you want to add the package prompt resource and click **OK**.

8 In the Outline view, navigate to the base package and expand **Resources**.

9 Right-click the custom package prompt and select **Properties**.

10 Next to the **Targets** field, click **Add**.

11 Expand the Package Catalog and select the base package to which you added the custom prompt, then click **OK**.

12 Specify the order in which you want the Driver Configuration Wizard to display the custom prompt. The Wizard displays prompts in ascending order starting from `0`.

13 Click **Prompts**.

14 Click **Add** to add each new prompt you want to include in the custom prompt Resource object. For information about adding new prompts, see "Adding Prompts" on page 225.

15 When finished adding prompts, click **Apply**.

16 Click **Prompt Transformation**. This window allows you to configure how you want to display the prompt in the Driver Configuration Wizard.

17 (Conditional) If you want to use a default prompt transform as the prompt transform for your custom prompt, click **Generate from template** and select the template you want to use, then click **OK**. Designer automatically populates the Stylesheet window with the selected template.

WARNING: When you generate the prompt transform from a template, Designer overwrites any XML currently in the Stylesheet window. If you have any previously-customized XML, ensure that you save the existing XML before clicking **Generate from template**.

---

18 Modify the default transform as necessary for your custom package prompt. For more information about default prompt transforms, see "Understanding Package Prompts" on page 204. For more information about transforms, see "Understanding Package Prompt Transformations" on page 208.

19 Click **Apply**.

20 Click **Target Transformation**. This window allows you to configure how you want to modify the target of the transform.

21 (Conditional) If you want to use a default target transform as the target transform for your custom prompt, click **Generate from template** and select the template you want to use, then click **OK**. Designer automatically populates the Stylesheet window with the selected template.

WARNING: When you generate the target transform from a template, Designer overwrites any XML currently in the Stylesheet window. If you have any previously-customized XML, ensure that you save the existing XML before clicking **Generate from template**.

---

22 Modify the default transform as necessary for your custom package prompt. For more information about default target transforms, see "Understanding Package Prompts" on page 204. For more information about transforms, see "Understanding Package Prompt Transformations" on page 208.

23 Click **OK**.

Editing Package Prompts

You can edit the properties of a Resource object to change the package prompts to meet your needs. You can add new prompts, edit the existing prompts, or add default values for the prompts that are displayed when the package is installed.

- "Adding Prompts" on page 225
- "Editing Existing Prompts" on page 225
- "Setting Default Values for the Prompts" on page 225

Adding Prompts

1 In the Outline view, right-click the Prompt Resource object in the package, then click **Properties.**

2 Click **Prompts**, then click **Add**. For more information about adding a GCV resource as a prompt, see "Global Configuration Value Definition Editor" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

3 Click **Finish** to save the changes and close the page.

Editing Existing Prompts

1 In Outline view, right-click the Prompt Resource object in the package, then click **Properties**.

2 Click **Prompts**.

3 Select the prompt, then click **Edit**.

4 Make the desired changes, then click **Finish**.

Setting Default Values for the Prompts

1 In the Outline view, right-click the Prompt Resource object in the package, then click **Properties**.

2 Click **Prompts**.

3 Specify the default value in each prompt, then click **Apply** to save the changes.

4 Click **OK** to close the Prompts page.

Creating Identity Vault and Driver Set Packages

When creating custom packages, you may determine that some of the content in your base and feature packages can be used at a higher level, in other drivers in the driver set or in the Identity Vault as a whole.

You can create common packages on driver sets and Identity Vaults and add libraries, policies, ECMAscript objects, GCVs, password policies, and other object types to those high-level packages. You can also add notification templates to an Identity Vault package.

To create an Identity Vault or driver set package, complete the following steps:

1 In the Package Catalog, right-click the package group where you want to create a new package and select **New Package**.

2 Specify a name, version number, and description for the package in the appropriate fields.

3 Specify a short name for the package in the appropriate field. OpenText Identity Manager and Designer display the specified short name when you open the package in a user interface. This name must be unique in the Identity Vault.

NOTE: The standard short name for a package is 12 characters long, separated into three sections of four characters: `[Vendor][Target system][What package does]`.

For example, if you have a common settings driver set package created by OpenText, the package short name could be `NTIQCOMMSTNG`. If you have an Identity Vault package created by OpenText that contains password synchronization notification templates, the package short name could be `NTIQPSYNNOTF`.

---

4 Click the **Type** drop-down menu and select **DriverSet** or **Identity Vault**, depending on the type of package you want to create.

5 Verify the package category and group are correct.

6 Click **Next**.

7 In the **IDM Compatibility** section, select the minimum and maximum versions of OpenText Identity Manager that this package is compatible with, then click **Next**.

8 Specify or modify the vendor information you want to include in the package, then click **Next**. You must specify the vendor name for the package.

9 Review the Summary page and click **Finish**.

10 (Optional) If you want to require a particular Identity Vault package be installed along with your driver set package, complete the following steps:

10a In the Outline window, expand the Package Catalog and navigate to the version of the driver set package you created in the preceding steps.

10b Right-click the driver set package and select **Properties**.

10c In the Properties window, click **Dependencies**.

10d Click the plus icon to and select the Identity Vault package object you want to add as a dependency.

NOTE: You can only add an Identity Vault packages as a dependency for a driver set package. You cannot set any type of package as a dependency for an Identity Vault package.

10e Click **OK**.

11 In the Modeler, right-click the Identity Vault or driver set, depending on the type of package you created, and select **Properties**.

12 In the Properties window, click **Packages** to install the package on the Identity Vault or driver set.

13 Click the plus icon to display the packages you can install.

14 Select the package you want to install and click **OK**.

15 Click **OK**.

16 Click **Finish**.

Creating Libraries

In order to add policies, style sheets, rules, or other objects to an Identity Vault or driver set package, you must first create a custom library on the Identity Vault or driver set, as appropriate. You then create the new objects in the library and add those objects to your Identity Vault or driver set package.

NOTE: You cannot add the library itself to the Identity Vault or driver set package.

For more information about working with libraries in Designer, see "Library Objects" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

To add and populate a custom library, complete the following steps.

1 In the Modeler, right-click the Identity Vault or driver set and select **New > Library**.

2 Specify a name for the new library and click **OK**.

3 Right-click the new library and select **New**, then select the type of object you want to add to the library. For information about adding objects to a library, see "Adding Policies to the Library Objects" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

4 After you add the new object, right-click the object in the Outline view and select **Add to Package**.

5 Select the Identity Vault or driver set package where you want to add the object and click **OK**.

NOTE: Only packages that are created in Designer are displayed in the list. Any packages that are imported into Designer are not displayed in the list.

6 Repeat Step 3 through Step 5 for each object you want to add.

7 (Optional) If your driver requires the objects included in the library, complete the following steps:

7a Right-click the library and select **Live > Deploy**.

7b Click **Deploy**.

7c Click **OK**.

Adding GCV Resource Objects

After you create an Identity Vault or driver set package, you can create and add new GCV objects to the package. To create and configure a GCV resource object, complete the following steps:

1. Install the feature package you want to use on your development driver. For more information about installing the development driver, see "Creating a Development Driver" on page 195.
2. In the Outline view, right-click the driver name and select **New > Global Configuration**.
3. Specify a name for the new GCV resource object and click **OK**.
4. In the Outline view, right-click the new GCV resource object and select **Add to Package**.
5. Select the feature package where you want to add the GCV resource and click **OK**.

   NOTE: Only packages that are created in Designer are displayed in the list. Any packages that are imported into Designer are not displayed in the list.

6. Right-click the GCV resource and select **Properties**.
7. Click **GCVs**.
8. Click **Add** to add a new global configuration value. For more information about adding a GCV, see "Global Configuration Value Definition Editor" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.
9. Click **Finish**.
10. Repeat Step 8 through Step 9 for each GCV you want to add.
11. Click **OK**.

Adding Notification Templates

In addition to libraries and GCVs, you can add notification templates to Identity Vault packages. Notification templates allow you to automatically send e-mail messages to users as part of a policy workflow.

For example, if you add a password-management feature to your driver where OpenText Identity Manager auto-generates a password for a user as soon as that user is provided with an account on your application, you need a notification template to e-mail that user their new password. For more information about creating and using notification templates, see "Setting Up E-Mail Notification Templates" on page 271.

To add a notification template to a package, complete the following steps.

1. In the Outline view, right-click **Default Notification Collection** and select **New Template**.
2. Specify a name for the new notification template and click **OK**.
3. In the E-Mail Template Editor, configure the notification template. For information about configuring notification templates, see "Setting Up E-Mail Notification Templates" on page 271.
4. When finished, close the template and click **Yes** to save the resource.
5. Right-click the template in the Outline view and select **Add to Package**.
6. Select the Identity Vault package where you want to add the object and click **OK**.

NOTE: Only packages that are created in Designer are displayed in the list. Any packages that are imported into Designer are not displayed in the list.

7 Repeat Step 1 through Step 6 for each notification template you want to add.

Creating Feature Packages

After creating a base package, you need to create the feature packages that users install with the base package. Feature packages contain the bulk of the actual content for a driver, including policies, GCVs, filters, and prompts.

Creating the content for a package is different than creating the package. This section explains how to create the package, then "Adding Content to Packages" on page 231 explains how to add the content to the package.

If you need several feature packages that cover a similar area of functionality, you can organize those packages using package groups. For example, when you install the LDAP driver using the LDAP Base package (NOVLLDAPBASE), the optional features listed do not display the name of each specific package by default but instead group features into the package groups Default Configuration, Entitlements, Password Synchronization, Data Collection, and Account Tracking. Users can then choose to install those optional features as a whole, rather than selecting a particular package.

NOTE:
- We recommend you create and configure mandatory feature packages sparingly. If a feature or resource is required for all installations of the driver, you should include the feature in the base package, instead.
- All packages must belong to a category and a group within that category. You cannot create a package outside of a package group.
- Feature packages should belong to the same package group and category as the base package to which they belong.
- When you create multiple feature packages, we recommend using package groups to organize packages by feature. This can make the structure of the different features more clear to the end user.

1 Right-click the package group where you want to create a new package and select **New Package**.

2 Specify a name, version number, and description for the package in the appropriate fields.

3 Specify a short name for the package in the appropriate field. OpenText Identity Manager and Designer display the specified short name when you open the package in a user interface. This name must be unique in the Identity Vault.

NOTE: The standard short name for a package is 12 characters long, separated into three sections of four characters: `[Vendor][Target system][What package does]`.

For example, if you have an Active Directory feature package created by OpenText related to email, the package short name could be `NTIQADIRBASE`.

4 Click the **Type** drop-down menu and select **Driver**.

5 Verify the package category and group are correct.

6 Click **Next**.

7 In the **IDM Compatibility** section, select the minimum and maximum versions of OpenText Identity Manager that this package is compatible with. The selected versions should correspond to the versions selected for the base package.

8 In the **Application Compatibility** section, select the minimum and maximum versions of the managed application that this package is compatible with. The selected versions should correspond to the versions selected for the base package.

9 Select one or more driver types in the **Available Driver Types** list with which you want the package to be compatible and use the right-arrow icon to move them to the **Supported Driver Types** list

---

NOTE: The package must support at least one driver type. Ensure you select the type of application you used when creating your development driver.

---

10 Click **Next**.

11 Specify or modify the vendor information you want to include in the package, then click **Next**. You must specify the vendor name for the package.

12 Review the Summary page and click **Finish**.

Configuring Mandatory and Optional Feature Packages

Feature packages can be mandatory or optional, depending on the functionality you want to provide. If you need a particular feature, you can configure that feature package to be mandatory, while leaving other, less-essential feature packages as optional.

You specify the mandatory and optional feature packages for a base package in the Configuration Wizard Properties page of the base package, using the XML tags `<mandatory></mandatory>` and `<optional></optional>`. This XML document configures how the Configuration Wizard displays features for the base package when you install the package on a driver.

1 In the Outline window, expand the Package Catalog and navigate to the version of the feature package you want to configure as mandatory or optional.

2 Select the feature package.

3 In the Properties view, find the Package Id field and copy-and-paste the package ID number into a text file.

4 Repeat Step 1 through Step 3 for each feature package you want to configure, saving all package IDs.

5 In the Designer Outline window, expand the Package Catalog and navigate to the version of the base package for which you want to configure sub-packages.

6 Right-click the base package and select **Properties**.

7 In the Properties window, click **Configuration Wizard**.

8 In the Configuration Wizard Feature Definition window, modify the XML to include all mandatory and optional feature packages, using the following XML structure:

```
<?xml version="1.0" encoding="UTF-8"?><features>
    <mandatory>
        <group display-name="Mandatory Package Group Name1"
expanded="false">
            <package display-name="Mandatory Package Name1"
id="PackageIDNumber1" selected="true"/>
        </group>
        <group display-name="Mandatory Package Group Name2"
expanded="false">
            <package display-name="Mandatory Package Name2"
id="PackageIDNumber2" selected="true"/>
        </group>
    </mandatory>
    <optional>
        <group display-name="Optional Package Group Name 1"
expanded="false">
            <package display-name="Optional Package Name1"
id="PackageIDNumber3" selected="true"/>
        </group>
        <group display-name="Optional Package Group Name2"
expanded="false">
            <package display-name="Optional Package Name2"
id="PackageIDNumber4" selected="true"/>
        </group>
    </optional>
</features>
```

Paste the copied package IDs into the XML as the values of your `id` fields. Each feature package must have a unique package ID.

You can have multiple groups within the `<mandatory>` and `<optional>` tags. If you want a package to be selected by default in the Configuration Wizard, ensure the value of the `selected` attribute is `true`.

NOTE: If there are no mandatory feature packages, use the XML tag `<mandatory/>`.

9 Click **OK**.

Adding Content to Packages

After you have created a package, you must add OpenText Identity Manager content to the package for the package to have value.

You can add different types of content to a package, including policies, ECMAScript objects, package prompt resources, and entitlements. For a full list of all types of content you can add to a package, see Table 6-1 on page 174.

IMPORTANT: You can only add content to a package you create. You cannot add content to a package you have imported unless you also have the Designer project in which the package was developed.

For more detailed information about adding GCVs, prompts, policies, and filter extensions to a feature package, see the following sections:

- “Adding GCVs to Feature Packages” on page 232
- “Adding Prompt Resources” on page 232
- “Adding Policies” on page 233
- “Adding Filter Extensions” on page 234
- “Adding Content to a User Application Driver” on page 235
- “Modifying Content of a User Application Driver” on page 235

To add content to a feature package, you must first install the package on the driver, add the content item to the driver, then add the configured content item to the package. You can then view the content item under the feature package in the Package Catalog. When users install the package, whatever language Designer is using is the language in which the package itself is installed.

Complete the following steps to install the package on the driver:

1. Verify you have a development driver installed. If not, follow the steps in “Creating a Development Driver” on page 195 to install a development driver.
2. Verify that you have created a feature package. Otherwise, follow “Creating Feature Packages” on page 229 to create a new feature package.
3. In the Modeler, right-click the development driver, then click **Driver > Properties**.
4. In the Properties window, click **Packages** to install the feature package on the driver.
5. Click the plus icon to display the packages you can install on the driver.

   The package list is initially filtered by driver types. To see all available driver packages, deselect **Show only applicable package versions**.
6. Select the feature package you want to install and click **OK**.
7. Click **OK**.
8. Specify configuration information for any prompts displayed in the Installation Wizard, then click **Next**.
9. Click **Finish** to install the package.

Adding GCVs to Feature Packages

As with Identity Vault and driver set packages, you can also add GCVs to a feature package. For information about adding GCVs to a package, see “Adding GCV Resource Objects” on page 228.

Adding Prompt Resources

To add package prompts to a feature package, complete the following steps:

1. Verify that you have created a feature package. Otherwise, follow “Creating Feature Packages” on page 229 to create a new base package.
2. Right-click the feature package in the package catalog and select **Generate Prompt Resource**.
3. Select the type of package prompt you want to configure:

Initial Settings: This option creates all of the default attributes required to create a driver object.

Upgrade Settings: This option creates a Resource object that contains style sheets that maintain the package settings so that they are not overwritten when the new package is installed. Select this option if the package you are creating is an upgrade to an existing package.

NOTE: You can only create one of each type of prompt for a particular package.

---

4 In the package catalog, expand the package version and Resources directory.

5 Right-click the new package prompt and select **Properties**.

6 Verify the type of package prompt.

7 Specify the order in which you want the Driver Configuration Wizard to display the current package prompt. The Wizard displays prompts in ascending order starting from `0`.

8 Verify the target displayed is correct for the package prompt. If you want to add the prompt to a different package, click **Add**, browse to the package, and click **OK**.

9 Click the Prompts tab. The Properties window displays what the current package prompt looks like in the Driver Configuration Wizard.

Adding Policies

To add policies to a feature package, complete the following steps:

1 Verify that you have created a feature package. Otherwise, follow "Creating Feature Packages" on page 229 to create a new base package.

2 In the Outline view, right-click the driver name and select **New > DirXML Script**.

3 Specify a name for the new policy and click **OK**.

4 In the Outline view, right-click the new policy and select **Add to Package**.

5 Select the feature package where you want to add the policy and click **OK**.

NOTE: Only packages that are created in Designer are displayed in the list. Any packages that are imported into Designer are not displayed in the list.

---

6 Double-click the policy and use the Policy Builder to add rules as necessary. For information about building policies in the Policy Builder, see "Managing Policies with the Policy Builder" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

7 Close the policy and click **Yes** to save the resource.

8 Repeat Step 5 through Step 7 for each policy you want to add.

Adding Filter Extensions

When you create a custom feature package, you should configure OpenText Identity Manager to allow data flowing through your environment to go through your new driver's workflow. For your driver and associated packages to process data, you must create a **filter**.

Filters act as gates to stop data going into or out of your driver. Filters allow you to specify criteria against which the driver matches any incoming our outgoing data and then executes a specified action. You can filter data on both the Publisher or Subscriber channels of your driver, or simply set up a filter that notifies you when an object is modified.

You should understand the types of data you want the driver with that package installed to process. You can then configure the specific subset of data you want to be processed or synchronized by the driver.

For example, you may want the driver to sync data regarding user objects. You can create a filter extension within your feature package that allows any data related to user objects through the workflow, while blocking any other type of data. If the Identity Vault sends an event about a group object to your driver, the filter sees that the event is not about a change to a user object and does not send the event through the driver workflow.

To create a filter, you must create a **filter extension resource** in your feature package and then deploy that package to a driver. For more information about filter extensions, see "Controlling the Flow of Objects with the Filter" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

Complete the following steps to create a filter.

1. Verify that you have created a feature package. Otherwise, follow "Creating Feature Packages" on page 229 to create a new base package.
2. In the Outline view, right-click the driver name and select **New > Resource**.
3. Specify a name for the new filter resource.
4. Click the **Content type** drop-down menu and select **application/vnd.novell.dirxml.filter-ext+xml**.

   NOTE: Filter extensions can only be created and modified when the Identity Vault is running in package development mode. You cannot perform these operations with development mode disabled.

5. Click **OK**.
6. In the Filter Editor, add and configure filters as necessary. For information about configuring filters in the Filter Editor, see "Controlling the Flow of Objects with the Filter" in *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.
7. Close the filter and click **Yes** to save the resource.
8. In the Outline view, right-click the new filter and select **Add to Package**.
9. Select the feature package where you want to add the filter extension and click **OK**.

   NOTE: Only packages that are created in Designer are displayed in the list. Any packages that are imported into Designer are not displayed in the list.

10. Repeat Step 2 through Step 9 for each filter you want to add.

Adding Content to a User Application Driver

You must add content to a User Application driver through a non-base package, as described in the following steps:

1. Create an Identity Vault in the package development mode.
2. Install the User Application driver with the latest User Application package.
3. Create a new User Application non-base package and install it on the User Application driver.
4. Configure custom objects such as roles, resources, PRDs, or categories and then add them to the non-base package.
5. Release the non-base package after you have completed configuring the custom objects.

You must install the non-base package only on a driver that already includes a User Application base package. After you have completed the preceding steps, the customizations that you made to the custom objects are preserved on further upgrades or downgrades of the non-base package.

Modifying Content of a User Application Driver

You must modify the existing content of a User Application driver through a non-base package, as described in the following steps:

1. Create an Identity Vault in the package development mode.
2. Install the User Application driver with the latest User Application package.
3. Create a new User Application non-base package and install it on the User Application driver.
4. Modify the existing Data Abstraction Layer objects and then add them to the non-base package.

IMPORTANT: Ensure that you add the entire object group to the non-base package because Designer does not allow you to add individual system objects. For example, to modify an Entity object, add the entire Entity group to the non-base package.

---

5. Release the non-base package after you have completed configuring the custom objects.

You must install the non-base package only on a driver that already includes a User Application base package. After you have completed the preceding steps, the customizations that you made to the custom objects are preserved on further upgrades or downgrades of the non-base package.

Copying Packages

In addition to creating a new package, you can also copy an existing package in the Package Catalog. Copying packages gives you the same content, but it contains a different global identifier. This allows you to create a new package based on the content of an existing package.

1. Verify that you have a package created with content. Otherwise, follow "Creating Feature Packages" on page 229 and "Adding Content to Packages" on page 231 to create a package with content.
2. Right-click the package in the package catalog you want to copy, then click **Copy Package**.

3 Use the following information to create a copy of the package:

- **Name:** Change the name of the package, if desired.
- **Short Name:** Change the unique short name for the package. This name must be unique in the Identity Vault.
- **Version:** Specify the package version you want to use. By default, the package version is set to 0.0.1.
- **Description:** Specify a description for the package.
- **Type:** This field cannot change. The package type is determined when you create a package, not when you copy a package.
- **Base Package:** If you want to use the copied package as a base package, select this option. If you leave this option cleared, Designer creates the copied package as a feature package.
- **Category:** Change the package category for this package, if desired.
- **Group:** Change the package group for this package, if desired.

4 Click **Next**.

5 Use the following information to define the package constraints:

- **IDM Compatibility:** Define the minimum and maximum versions of OpenText Identity Manager that the package supports.
- **Application Compatibility:** Define the minimum and maximum versions of the managed application that the package supports, if applicable.
- **Driver Type:** Select the drivers that the package supports, if applicable.

6 Click **Next**.

7 Use the following information to define the vendor of the package:

- **Vendor Name:** Specify the vendor name. If this package is for internal consumption, specify the name of your company.
- **Vendor Address:** Specify the address for the vendor or your company.
- **Vendor URL:** Specify the URL of the vendor or your company.
- **Vendor eMail:** Specify an e-mail for the vendor or your company.
- **Contact Name:** If there is a specific contact person for this package, specify their name.
- **Contact eMail:** If there is a specific e-mail address for the contact person, specify it in this field.

8 Click **Next**.

9 Review the summary of the new package version, then click **Finish**.

The copy of the package is created in the package catalog under the specified category and group. You can now build and release your package.

NOTE: When users install the copied package, the package uses the language used by Designer when the package was copied.

Building Packages

After you have created a custom package, you can build the package as a .jar file and prepare the file for consumption by other users.

1. In the Outline window, expand the Package Catalog and navigate to the version of the package you want to build.
2. Right-click the package and click **Build**.
3. Click **Browse**, then browse to and select the directory where you want to build the package.
4. Click **OK** twice.
5. Review the summary information, then click **OK**.
6. (Optional) After you build the package, provide the package to your QA team to verify, if appropriate. If the QA team finds any issues with the package, create a new version of the package to fix the bug. For more information about creating a new version of a package, see "Versioning Packages" on page 237.

Versioning Packages

You can create a new version of a package to provide bug fixes or enhancements to released packages. Versioned packages contain the same unique global identifier to support upgrading and downgrading package installations.

The version of a package consists of four parts separated by dots: `[Major Version][Minor Version][Patch Version][Package Creation Time Stamp]`. The version number parts should be used as follows:

- **Major Version:** You should increment the major version if you introduce a major feature in the new version of a package.
- **Minor Version:** You should increment the minor version if you introduce a minor or small feature in the new version of a package.
- **Patch Version:** You should increment the patch version if you make a small modification to a package.
- **Package Creation Time Stamp:** Designer automatically adds the time stamp when you create a new package and updates the time stamp each time you build the package. When you release a package, the time stamp is fixed.

To create a new version of a package:

1. In the package catalog, right-click the package you want to version, then click **New Package Version**.
2. Set the version of the package higher than the current version. All of the other fields stay the same when you are changing the version.
3. Click **Next**.
4. Modify the package constraints, if necessary, then click **Next**.
5. Modify the vendor information, if necessary, then click **Next**.
6. Review the summary of the new package version, then click **Finish**.

The new package with the new version number is created in the package catalog. You can now build and release your package. When users install the package, what ever language Designer is using, this is the language that the package is installed in.

Localizing Packages

You can localize the prompts and strings included in the custom packages you create. This allows you to provide the same package in multiple languages. Designer generates a localization property file that contain the strings that you can have localized.

NOTE: When you install a package on a driver, Designer displays the package prompts in the language in which Designer is open, if that localization property file is available.

Designer uses specific language codes to determine the language of a property file. For example, if you localize the English-language property file `NETQEDIRCFG_2.0.0.20120905154808_en.properties` in Spanish, the localized property file name should be `NETQEDIRCFG_2.0.0.20120905154808_es.properties`.

The following table provides the localization language codes available in Designer:

Language	Language Code
Japanese	_ja
Chinese Simplified	_zh_CN
Spanish	_es
French	_fr
Portguese Brazil	_pt_BR
Italian	_it
Chinese Traditional	_zh_TW
German	_de
English	_en
Dutch	_nl

To localize a package, complete the following steps:

1. In the Outline view, right-click the package in the package catalog, then click **Localization** > **Generate Property File**.
2. Click **Browse**, then browse to and select the directory where you want to store the property file.
3. Click **OK**.
4. Repeat Step 1 through Step 3 for each package you want to localize.
5. Take the property files and have them localized.
6. After the property files are localized, add the appropriate language code to the end of the file name.

7 Place the localized property files into a separate localization directory on the machine that is running Designer.

8 Open your project, then right-click the package in the package catalog.

9 Click **Localization** > **Import Property Files**.

10 Click **Browse**, then browse to the directory that contains the localized properties files.

11 Click **OK** three times.

12 To verify that you correctly localized the package properties, right-click the package and select **Properties**.

13 Click **Languages**. The Properties window displays all the languages in which the package is available.

14 Click **OK**.

You can now re-build and release your package.

Adding and Configuring Licenses

When developing packages to release to other users or to the public at large, you may need to include a license file with your released and published package. A license file is an HTML file that Designer displays when the user installs a new package.

You can either use one license as a default for all custom packages in your Designer environment or add licenses on a package-by-package basis. You can add a localized license for any of the languages listed in "Localizing Packages" on page 238.

NOTE: You do not need to add a license to a package for that package to function properly.

To add and configure licenses for your custom packages:

1 Obtain an HTML-format license file from the proper authorities in your company.

2 (Optional) If you want to use the license as the default for all packages you create, complete the following steps:

2a Click **Windows > Preferences**.

2b Click **NetIQ > Package Manager > License Defaults**.

2c Click **Browse**.

2d Click the browse button and navigate to the location of the license file you want to use as the default license.

2e Click **OK**.

2f Click the **Language** drop-down menu and select the appropriate language.

2g Click **Import**.

2h Click **OK**.

3 (Optional) If you want to use the license for a specific package, complete the following steps:

3a In the Outline view, right-click the package in the package catalog to which you want to add a license and select **Properties**.

3b Click **License**.

3c Click **Browse**.

3d Click the browse button and navigate to the location of the license file you want to use as the default license.

3e Click **OK**.

3f Click the **Language** drop-down menu and select the appropriate language.

3g Click **Import**.

3h Click **OK**.

Releasing and Publishing Packages

After you have finished developing and testing your custom package and localizing any necessary strings or prompts, you can release and publish the package. When you release and publish the package, other users can then use your package in their own OpenText Identity Manager environments.

You can publish the packages to a server and have users configure Designer to point to that server for package updates. You can specify a Web server (`http://`), FTP server (`ftp://`), or file server, as necessary for your environment. Users can then configure Designer to go to that location to check for package updates.

NOTE: ◆You can only publish a package to a location on the local system on which you are using Designer.

- Only packages that have been built and released can be published.

WARNING: After you release and publish a package, it becomes read-only. You cannot make any further modifications to the package.

---

1 In the Outline view, right-click the package in the package catalog you want to release, then click **Build**.

2 Click **Browse**, then browse to and select the directory where the package will be built and released.

3 Click **OK**.

4 Select **Release Package** and click **OK**.

5 Review the summary information, then click **OK**.

6 Right-click the built and released package, then click **Publish**.

The **Publish** option is not available until you have released the package.

7 In the **Publish Directory** field, click **Browse**, then browse to and select the Web server directory where you want to place the published package.

8 Click **OK**.

9 In the **Build Directory** field, click **Browse**, then browse to and select the directory where you built the package.

10 Click **OK** twice.

Designer stores the published package in the specified location on your Web, FTP, or file server. You can then configure Designer to check that location when checking for package updates.

1. Launch Designer.
2. From Designer's main menu, click **Windows > Preferences.**
3. Click **NetIQ > Identity Manager** and select the **Updates** tab.
4. Click the plus icon.
5. Specify a name for the Vendor and the URL for the Web, FTP, or file server, the click **OK.**
6. Click **OK** to close the Preferences window.

Best Practices for Package Development

We recommend that you try to adhere as closely as possible to the following best practices when developing custom packages:

- "Creating Packages" on page 241
- "Naming Packages" on page 241
- "Package Versioning" on page 242
- "Defining Package Relationships" on page 242
- "Adding Weights to Packages" on page 242
- "Documenting Packages" on page 243
- "Naming Package Items" on page 243
- "Reusing Package Content" on page 243

Creating Packages

- Do not create objects in a custom base package. A base package should be as lean as possible and should contain only the following:
  - Prompts
  - Initial settings
  - Information the base package's relationship to other packages
- If you have objects that are used by multiple drivers, store those items in a driver set package. You can create a driver set package, then store any often-reused objects in the package where any driver in the driver set can access the objects.

Naming Packages

- The standard package name is separated into two sections: `[Package Group] [Package Type]`. For example, if you have a base package for MySQL, the package name could be `MySQL Base`.

- Short names must be unique and cannot be longer than 12 characters.
- The standard short name for a package is separated into three sections of four characters: `[Vendor][Target system][What package does]`. For example, if you have a base Active Directory package created by OpenText, the package short name could be `NTIQADIRBASE`.

Package Versioning

- When creating a brand-new package, we recommend you begin numbering the package at version 0.0.1. After you finish creating and testing the package and are ready to release, then you can change the version to 1.0.0.
- Before you provide a custom package to a customer or other user, ensure you release the package. This helps ensure that if the user modifies the package, you do not have two different packages with different content but the same version number.
- You should release only the package with the most recent time stamp.

Defining Package Relationships

- You should configure Package A to be dependent upon another package, Package B, in the following situations:
  - One of the policies in Package A is dependent on a package item in Package B. This includes policies, GCVs, notification templates, and ECMAScripts.
  - Package A depends on some functionality included in Package B. For example, the Active Directory Password Sync package depends on the common password sync package, which defines all the necessary ECMAScript functionality.
- A mandatory feature relationship is a hard-coded dependency. You should avoid using mandatory features where possible.
- Instead, we recommend you configure any feature packages to be optional and then selected by default, using the `selected` XML attribute. Users can then deselect a feature if they do not want to install that feature. For information about configuring mandatory and optional feature packages, see “Configuring Mandatory and Optional Feature Packages” on page 230.

Adding Weights to Packages

We recommend that you use policy weights or the First/Last option while assigning linkages to policies in a package. Otherwise, policies without weights are moved after policies with weights and randomly sorted in packages.

If you use the policy weights option, ensure that the weights of adjacent policies vary significantly. As a best practice, use a policy weight ending with 5 or 0. This will ensure you have enough unique numbers for assigning weights to policies while adding them to packages.

Documenting Packages

When you create a new version of a custom package, you should use the package Readme to provide customers and users information about any changes from previous versions.

To add change information to a package Readme, right-click the version of the package in the Outline view and select **Properties**. Click **Readme**, then click **Append Package Change Log** to include any changes made since the previous version of the package. Click **OK** to exit.

Naming Package Items

Policies, Entitlements, ECMAScripts, and XSLTs: The standard name for these types of package items consists of four parts separated by hyphens: [Package Short Name]-[Channel Name (Optional)]-[Policy Set and Item Type]-[Item Name]. The item name parts should be used as follows:

- **Package Short Name:** This part should specify the short name of the package to which the item belongs.
- **Channel Name:** This part should specify if the item belongs to either the Publisher (`pub`) or Subscriber (`sub`) channel. If the item does not belong to either channel, do not include this part in the item name.
- **Policy Set and Item Type:** The first one or two characters of this part should refer to the policy set to which the item refers, including input transformation (`ip`), event transformation (`et`), creation (`c`), or matching (`m`). The last character in this part should be the item type, including policy (`p`), entitlement (`e`), ECMAScript (`c`), or XSLT (`s`).
- **Item Name:** This part should specify the job done by the package item.

For example, the name of a policy in an OpenText eDirectory package that belongs to the Publisher channel could be `NOVLEDIRATRK-pub-ctp-WriteAccountsOnAdds`.

Filters, Schema Maps, and Global Configuration Values: The standard name for these types of package items consists of two parts separated by hyphens: [Package Short Name]-[Item Type]. The first part should specify the short name of the package to which the item belongs. The second part should specify the type of the item, whether filter (`Filter`), schema map (`smp`), or global configuration value (`GCVs`).

For example, the name of a filter in an LDAP package could be `NOVLLDAPENT-Filter`.

WARNING: You can only specify a name with a maximum number of 64 characters for any object in a package. If you add an object with a name that is 65 or more characters long to a package, you cannot deploy the object.

Reusing Package Content

- If a package can be used by all driver sets in the Identity Vault, set the package type as `Identity Vault` when you create the package. For example, if you create a default notification template package, you must create that package as an Identity Vault package.

  For more information about creating Identity Vault packages, see "Creating Identity Vault and Driver Set Packages" on page 226.

- If a package can be used by all drivers in a particular driver set, set the package type as `DriverSet` when you create the package. For example, if you create a common settings package, you can create that package as a driver set package.

  For more information about creating driver set packages, see "Creating Identity Vault and Driver Set Packages" on page 226.

8 Managing the Flow of Data

Designer allows you to manage how the data flows between the Identity Vault and the managed systems. You can see how the data flows between all of the managed systems, make changes as needed, create reports about the data, and view the flow of passwords between the systems.

The Dataflow view and the Dataflow editor manage the data. The Dataflow view displays the flow of data in the Modeler per driver. The Dataflow editor displays a more granular view.

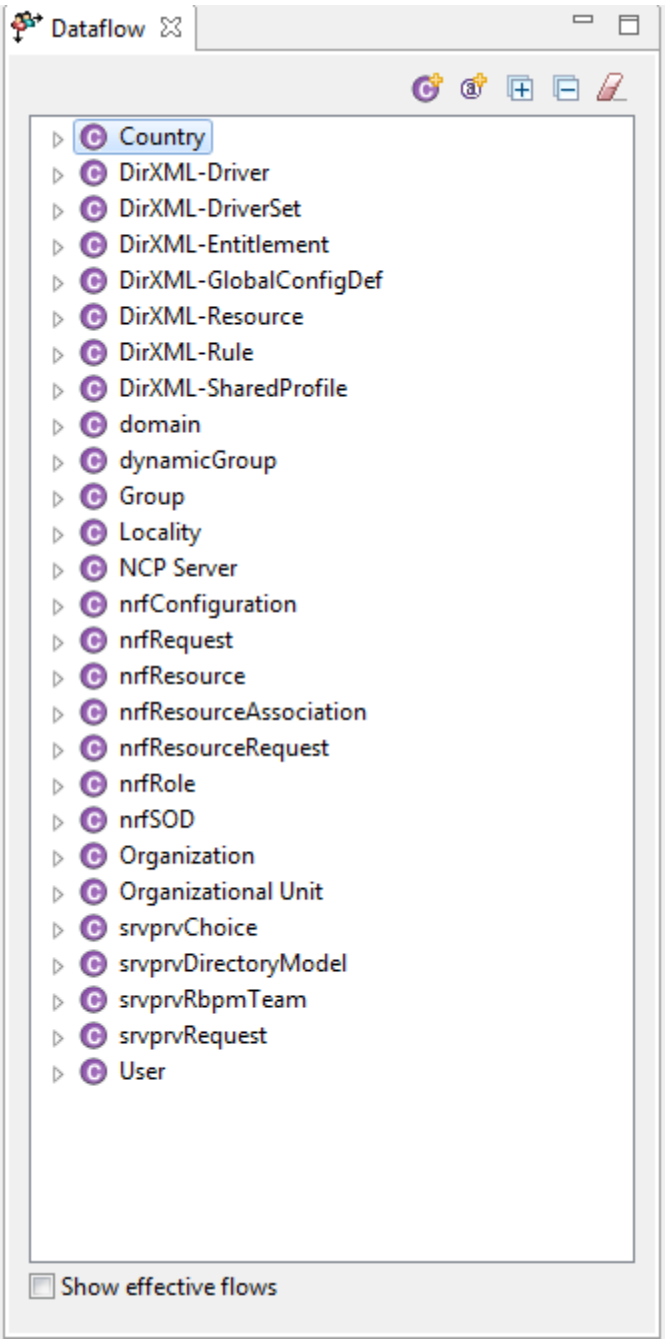
- "The Dataflow View" on page 245
- "The Dataflow Editor" on page 250
- "Adding Items in the Dataflow Editor" on page 257
- "Removing Items from the Dataflow Editor" on page 260
- "Editing Items" on page 260
- "Generating HTML Reports" on page 264
- "Integrating Passwords" on page 265

The Dataflow View

The Dataflow view displays a toolbar in the upper right corner of the view. For information about the icons in this toolbar, see "The Dataflow View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

The following figure illustrates the Dataflow view. You can use it to control the flow of data between the Identity Vault and managed systems. The Modeler displays the dataflow.

Figure 8-1 *The Dataflow View*



- "Accessing the Dataflow View" on page 247
- "Flow Arrows in the Modeler" on page 247
- "Viewing How Attributes Are Synchronized" on page 249
- "Changing the Data Flow" on page 249

Accessing the Dataflow View

If you have closed the Dataflow view, you can access it by selecting **Window** > **Show View** > **Dataflow**.

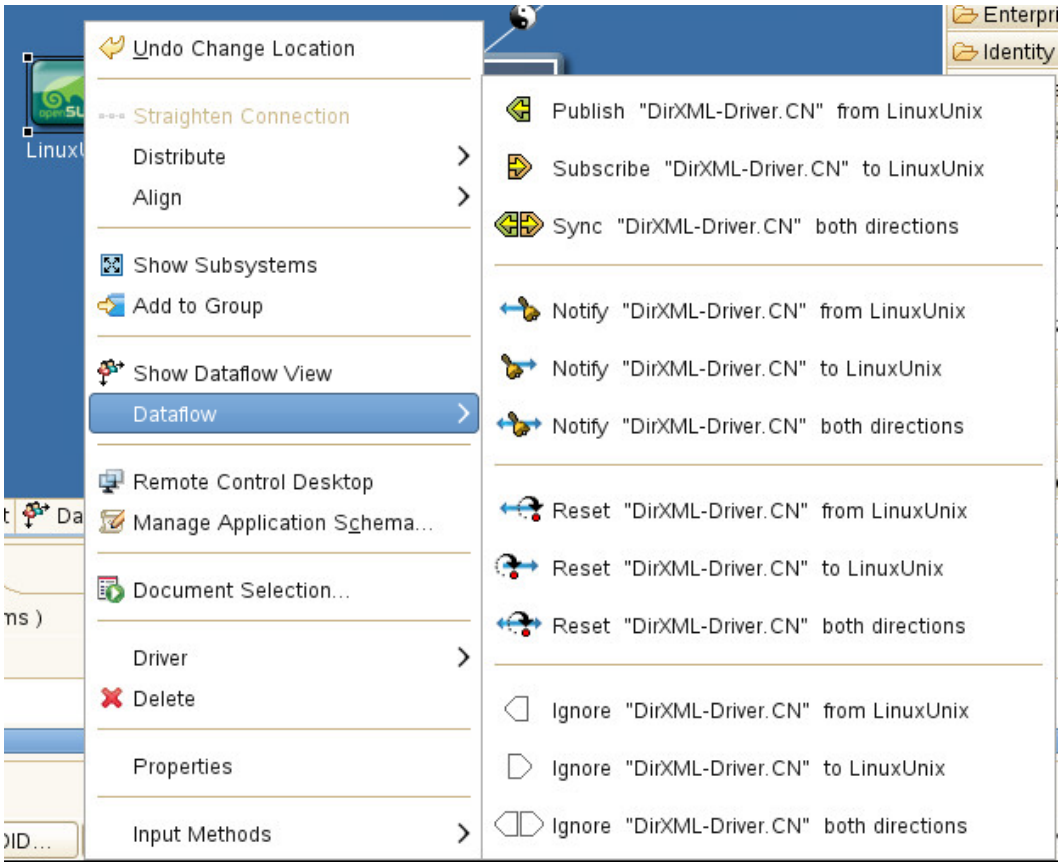
If the Dataflow view is blank and no project is displayed in the Modeler:

1. Expand a project in the Project view.
2. Open the project by double-clicking **System Model**.

   Objects and icons appear in the Dataflow view.

If you want to change how the data flows from the Modeler:

1. Right-click a driver or application in the Modeler.

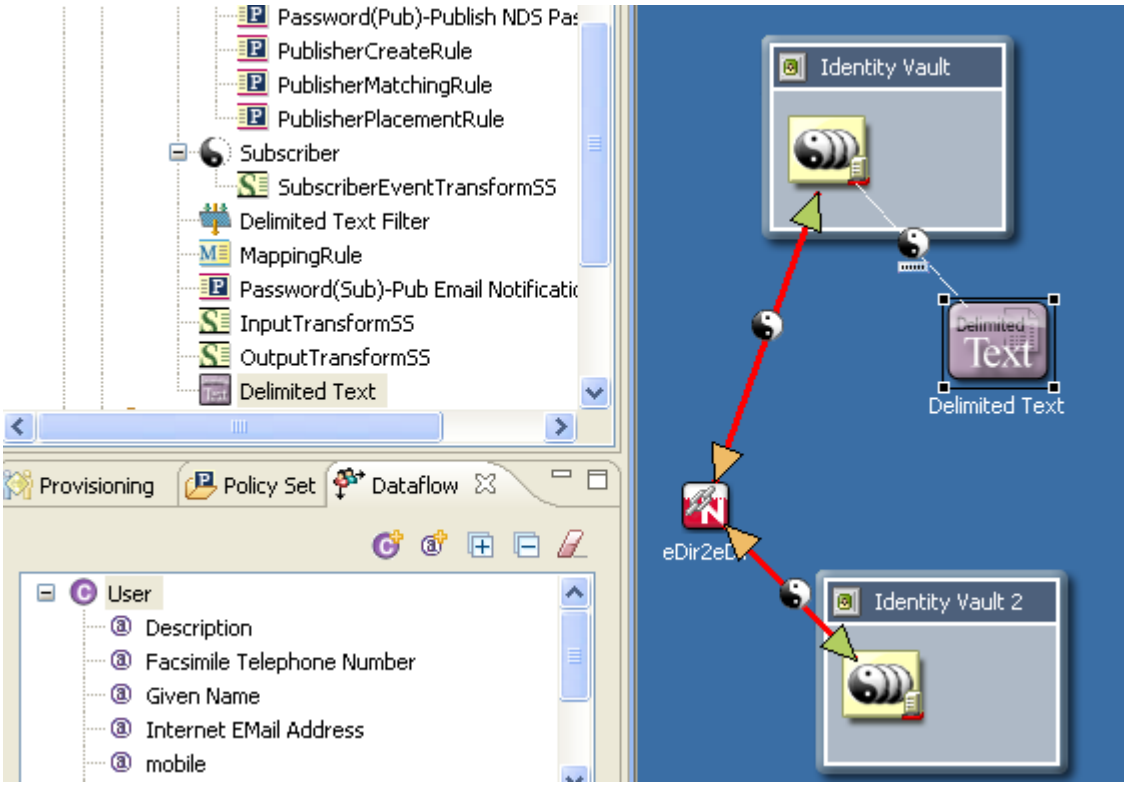


2. Select **Dataflow**, then select how you want the data flow to change.

Flow Arrows in the Modeler

When the Dataflow view opens, it automatically reads the filters and shows the classes and attributes. If a filter with classes and attributes doesn't exist, you can create one.

Figure 8-2 *Flow Arrows in the Modeler*



As you select a class or attribute in the Dataflow list, the appropriate driver lines are highlighted in red in the Modeler. Icons enable you to see Sync, Notify, Reset, and Ignore filter settings all at the same time.

Table 8-1 *Dataflow Icons*

Icon	Description
	Green arrow: the Publisher channel is synchronized.
	Orange arrow: the Subscriber channel is synchronized.
	Bell: the attribute is set to Notify.
	Reset arrow: the attribute is set to Reset.
No icon	The attribute is set to Ignore.

The color coding matches the Dataflow icons in the Filter editor and the Dataflow editor.

Viewing How Attributes Are Synchronized

To view whether attributes are synchronized or whether they will be notified, select **Show effective flows**. When you select this check box, the synchronize arrows do not show if the parent class is not set to synchronize. Therefore, you view an accurate diagram of actual flows.

However, if you want to view how attributes are configured to synchronize, regardless of the parent class, deselect **Show effective flows**. The synchronize arrows indicate which items are synchronized.

If you select an attribute that cannot synchronize (whether or not **Show effective flows** is selected), you see a **Blocked** warning in the upper left. This warning indicates that this attribute cannot be synchronized or notified because the parent class is not synchronized.

To view an explanation, mouse over the **Warning** icon.

Changing the Data Flow

You can change how the data flows for classes and attributes from the Dataflow view.

To change the flow for a class:

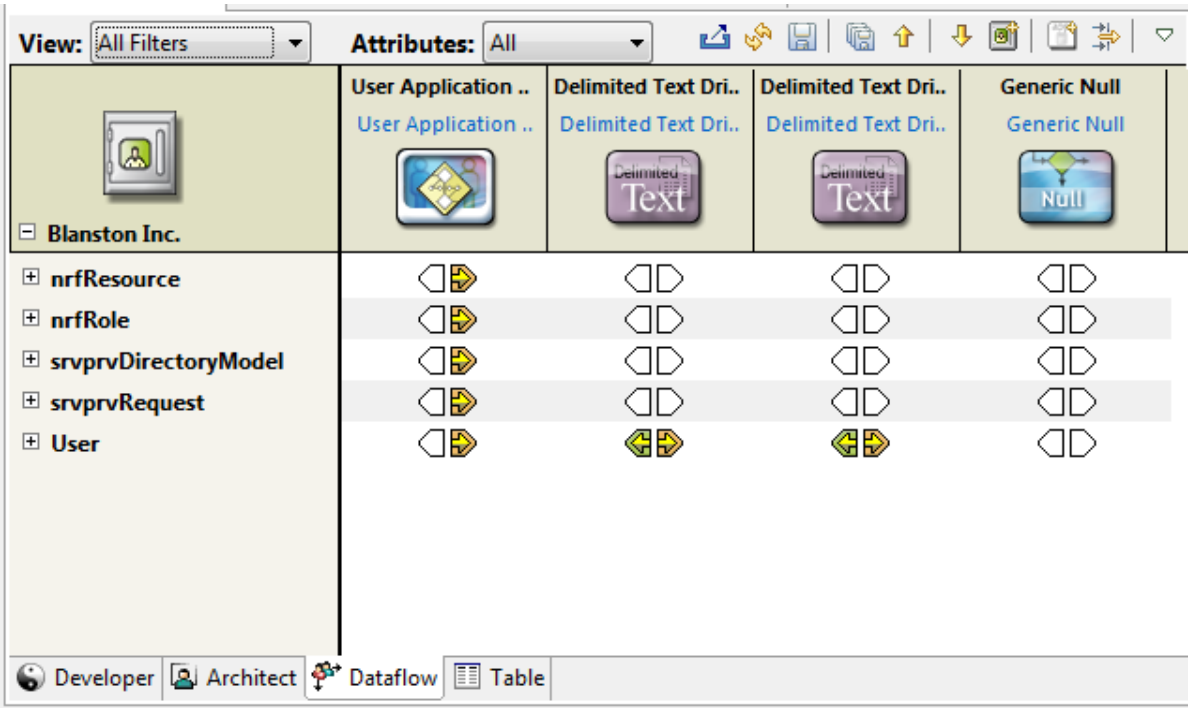
1. Select a class in the Dataflow view.
2. Right-click a driver line in the Modeler.
3. Select **Dataflow**.
4. Select the option to change the data flow for the class.

To change the flow for an attribute:

1. Select an attribute in the Dataflow view.
2. Right-click a driver line in the Modeler.
3. Select **Dataflow**.
4. Select the option to change the data flow for the attribute.

The Dataflow Editor

Figure 8-3 *The Dataflow Editor*



The Dataflow editor enables you to do the following:

- Use filters to display how data flows between all systems and Identity Vaults.
- View how passwords flow from each server.
- Generate reports of the data.

When object additions, deletions, changes, and selections synchronization occur, the Dataflow editor synchronizes with the Modeler and the Outline view.

To access the Dataflow editor, click the **Dataflow** tab.

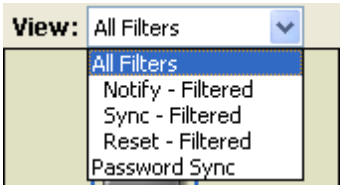
To adjust the area for the Identity Vaults, move the slider bar. This setting persists and is restored the next time you run the editor.

- "Filtering Views" on page 251
- "Filtering Identity Vaults and Applications" on page 252
- "Pinning the Identity Vault" on page 253
- "Expanding and Collapsing the Identity Vault" on page 254
- "Switching to an OpenText eDirectory Tree Icon" on page 256
- "Viewing an eDir-to-eDir Driver" on page 256
- "Keyboard Support" on page 257

Filtering Views

By default, the Dataflow editor shows all dataflows. The View drop-down list (in the upper left corner of the Dataflow editor, not in the Dataflow view), enables you to view notification, synchronization, reset, or Password Sync information. These filtered views do not allow you as much editing capability as the main view, but just what is necessary in that filter. For example, you can't add attributes, vaults, or applications, because by default they would not appear in the filter.

Figure 8-4 *Options to Filter Views in the Dataflow View*



- "Using the All Filters View" on page 251
- "Synchronizing Passwords" on page 251

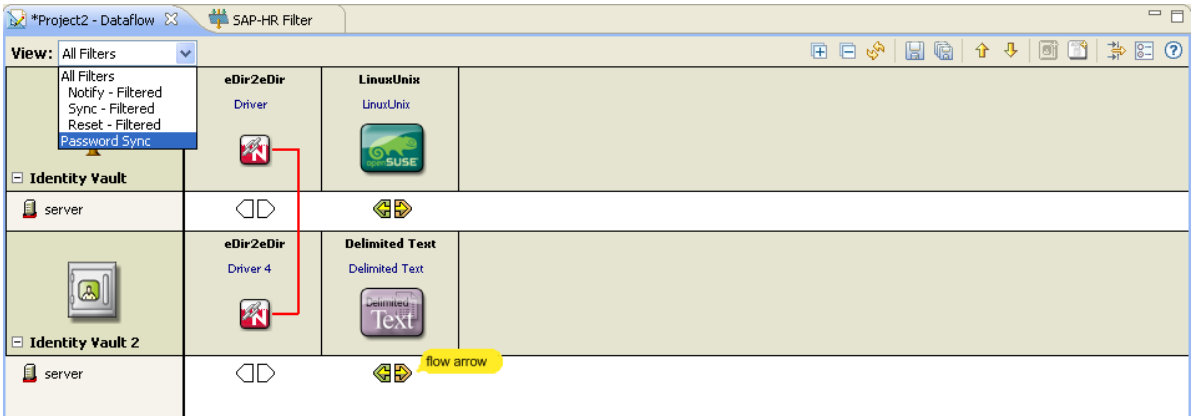
Using the All Filters View

If you are in the All Filters view, you can further filter with the **Attributes** list. Because the Dataflow editor provides non-filter attributes, you can choose to view regular filter-based attributes, non-filter attributes, or both.

Synchronizing Passwords

The Password Sync view enables you to see and edit how all passwords flow in the project. Designer displays the information about a per-server basis and shows how passwords flow among all of the applications.

Figure 8-5 *The Password Flow*



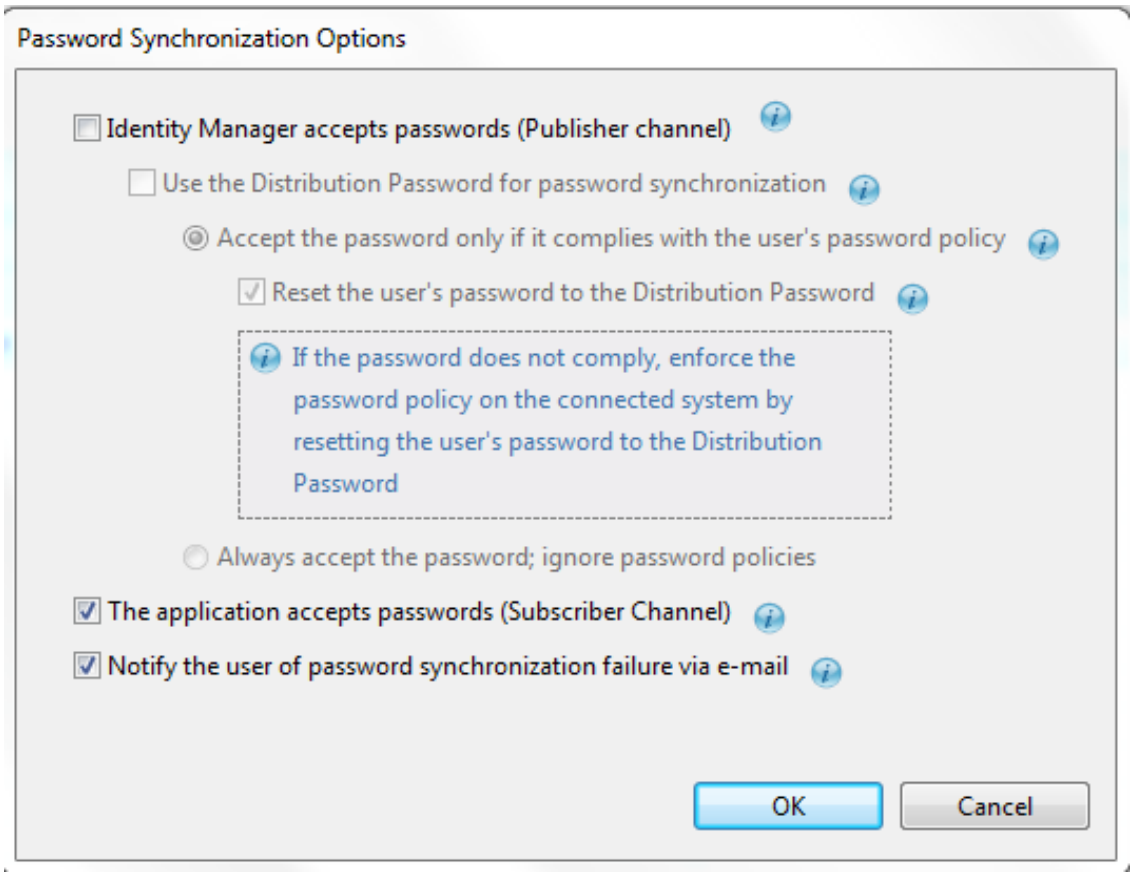
To edit the password flow:

1. Select **Password Sync** in the View filter.
2. Double-click the flow arrow.

You can also right-click, then select **Password Synchronization**.

3 Edit the password synchronization options.

For more information about password synchronization, see the *OpenText™ Identity Manager CE 24.4 (v4.10) Password Management Guide*.

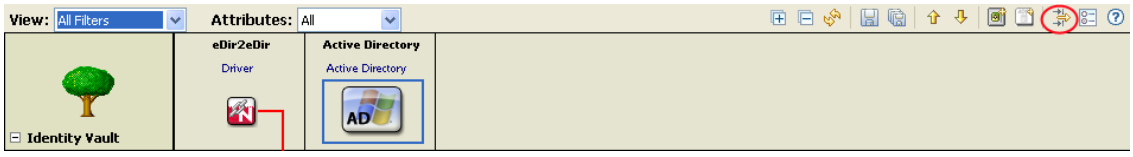


4 Click **OK**.

Filtering Identity Vaults and Applications

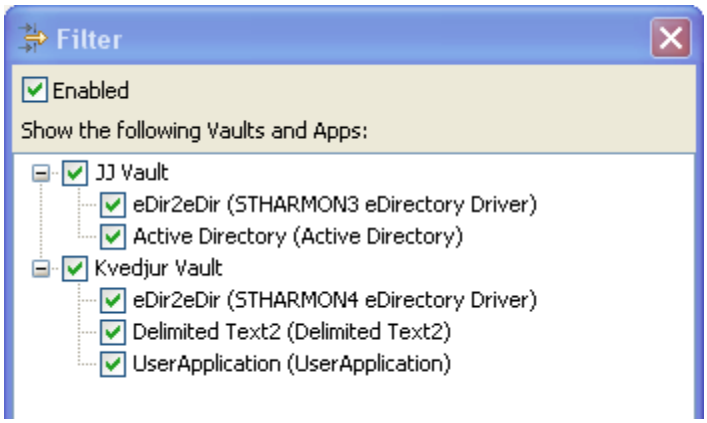
You can select the Identity Vaults and applications that you want to view in the editor.

1 In the Dataflow editor, click the **Filter View** icon.



2 Select **Enabled**.

The Identity Vaults and applications that you select here are included in the HTML reports. For more information, see "Generating HTML Reports" on page 264.

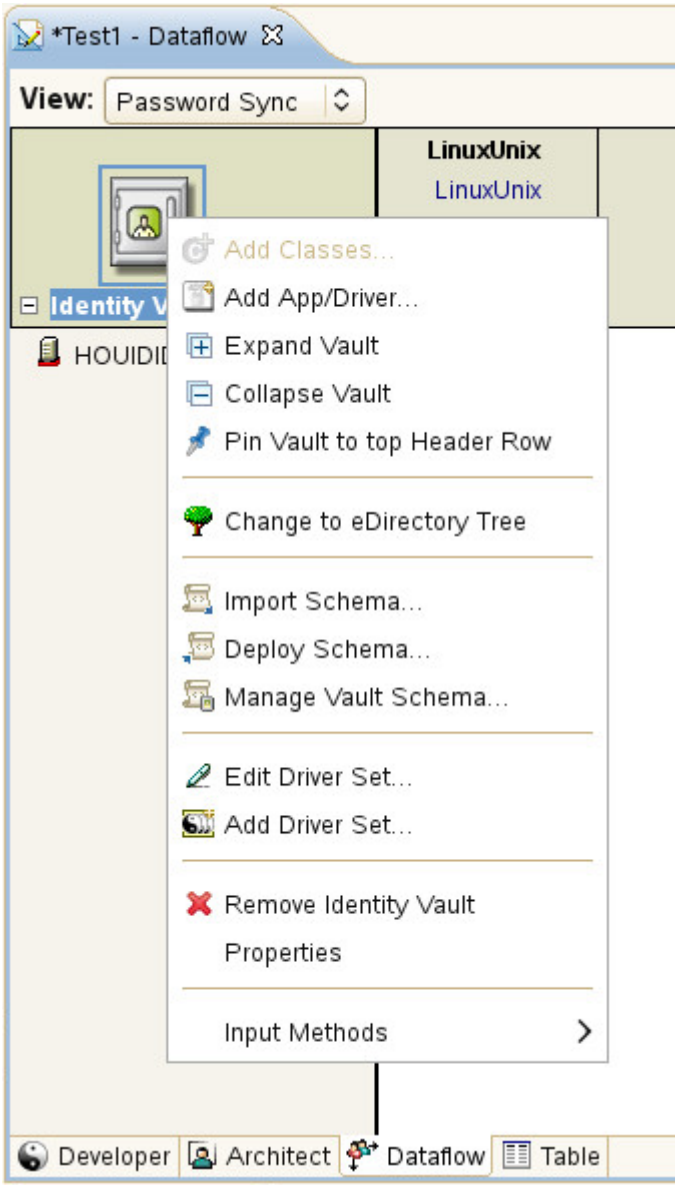


You can scroll and resize the dialog box. Also, you can interact with the Dataflow editor in the background, in any mode. This is convenient if you want to scroll a different section into view while this dialog box is up.

Pinning the Identity Vault

To change the scope of the editor to show a single Identity Vault, right-click the vault, then select **Pin Vault to Top Header Row**.

Figure 8-6 *Pinning an Identity Vault*



With a medium or large-sized project, the dataflow table can contain hundreds of rows and thousands of items. If you have multiple vaults and want to narrow the scope to more easily edit a vault without excessive scrolling, you might want to pin a vault. When an Identity Vault is pinned, a pin icon displays in the upper right corner.

To unpin the vault, right-click the Identity Vault, then select **Unpin Vault from Top Header Row**.

Expanding and Collapsing the Identity Vault

- "Expanding an Identity Vault" on page 255
- "Expanding All Identity Vaults" on page 255
- "Expanding Classes" on page 256

Expanding an Identity Vault

When the editor first loads, all vaults are expanded at the top level by default.

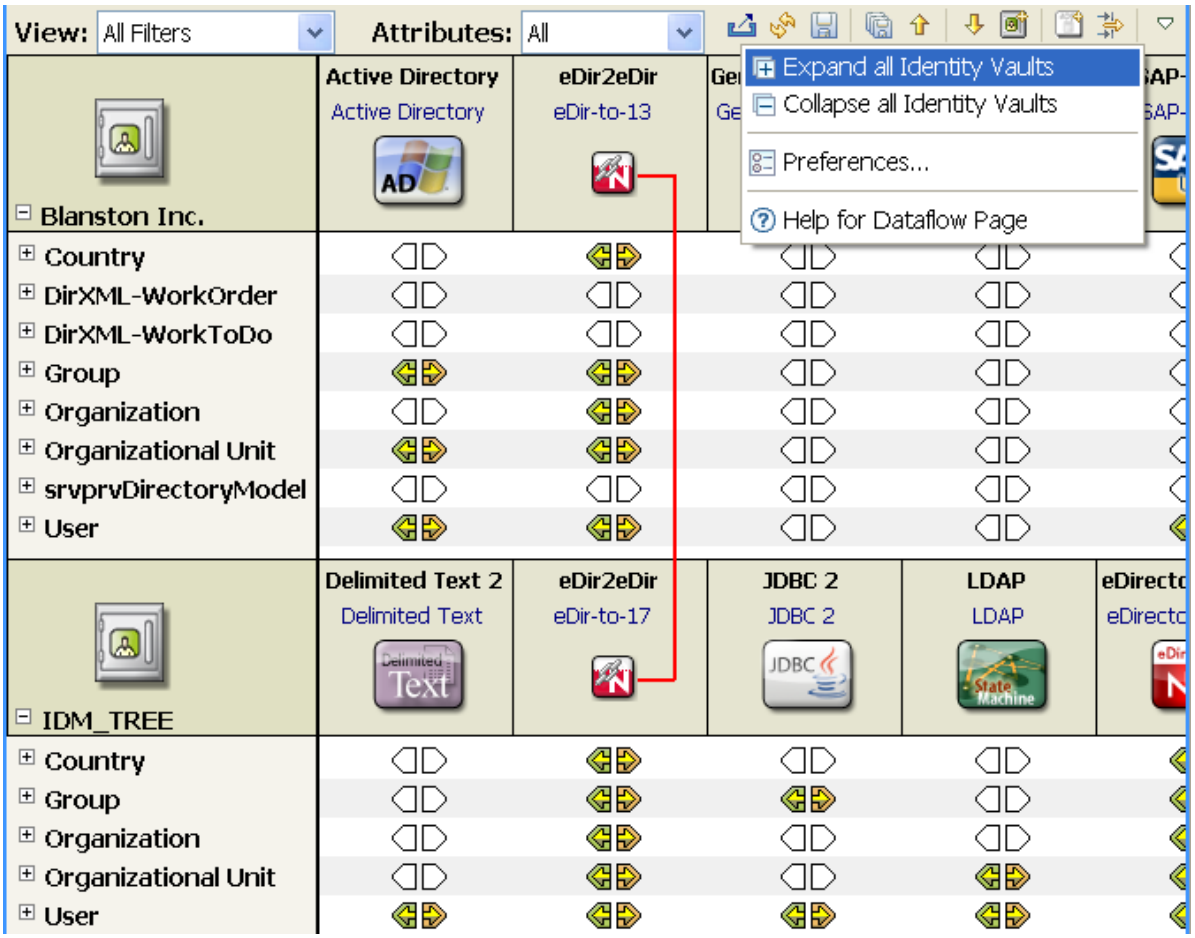
To collapse or expand the list of classes and attributes in an Identity Vault, do one of the following:

- Click the - or + icon below the Identity Vault icon.
- Select the Identity Vault, then press the Right-arrow key to expand the information, or press the Left-arrow key to collapse the information.

Expanding All Identity Vaults

To expand or collapse the list of classes and attributes for all Identity Vaults, click **Expand all Identity Vaults** or **Collapse all Identity Vaults** from the drop-down on the toolbar.

Figure 8-7 *Select to Expand or Collapse All Identity Vaults*



Expanding Classes

To view all attributes in a class, select the class, then press the Right-arrow key. To collapse the list of attributes, press the Left-arrow key.

To view all classes and attributes in an Identity Vault, right-click the Identity Vault icon, then select **Expand Vault.** To list just classes in an Identity Vault, right-click the Identity Vault, then select **Collapse Vault**.

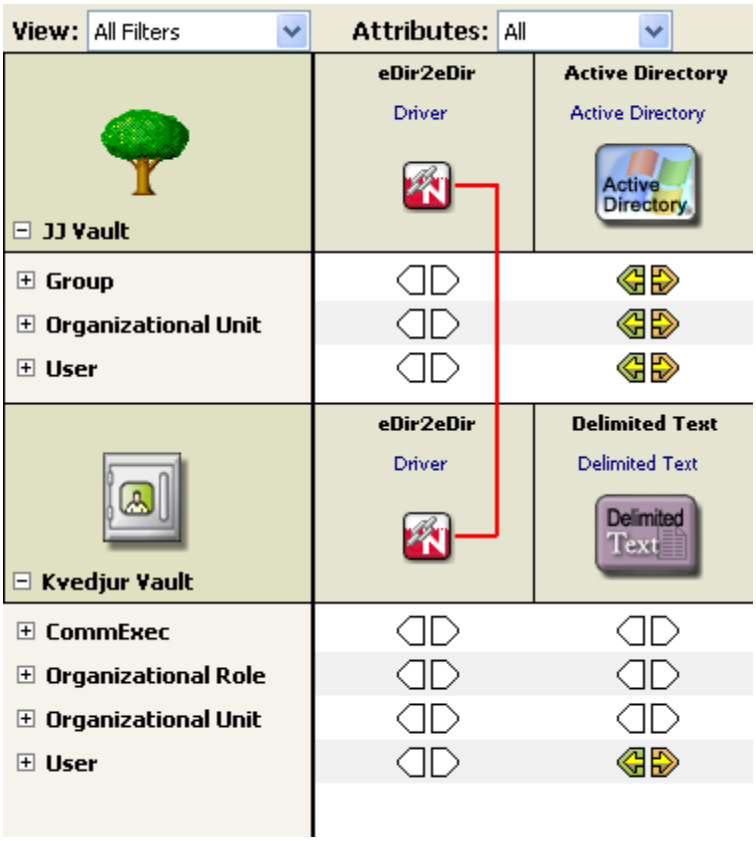
Switching to an OpenText eDirectory Tree Icon

To switch from an Identity Vault icon to an OpenText eDirectory tree icon, right-click the Identity Vault, then select **Change to eDirectory Tree.**

Viewing an eDir-to-eDir Driver

You can easily view both ends of an eDir-to-eDir connection so that you can configure the dataflows on both sides. Designer automatically detects the two OpenText eDirectory applications and aligns them in the same table column. A red line connects them.

Figure 8-8 *An eDir-to-eDir Connection*



Keyboard Support

You can navigate by using the Up-arrow, Down-arrow, Left-arrow, and Right- arrow keys as well as PageUp, PageDown, Home, and End. In addition, you can navigate from one Identity Vault to another by clicking the up-arrow or down-arrow on the toolbar.

Adding Items in the Dataflow Editor

- "Adding an Identity Vault in the Dataflow Editor" on page 257
- "Adding a Driver in the Dataflow Editor" on page 257
- "Adding an Application in the Dataflow Editor" on page 257
- "Adding Classes and Attributes" on page 258
- "Adding Non-Filter Attributes" on page 259

Adding an Identity Vault in the Dataflow Editor

To add an Identity Vault, click the **Add Identity Vault** icon on the toolbar.

To configure the Identity Vault, double-click it.

To delete an Identity Vault, select it, then press the Delete key.

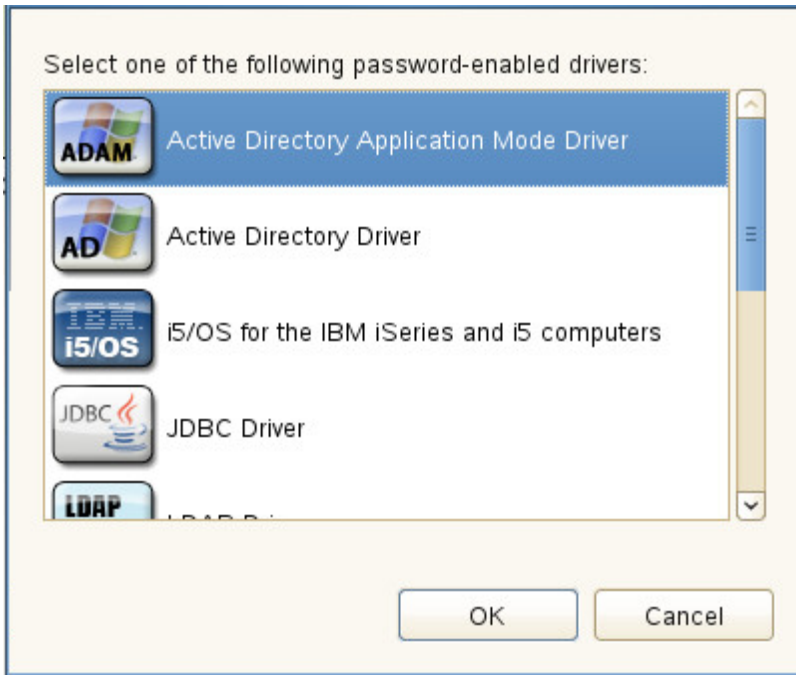
Adding a Driver in the Dataflow Editor

To add a driver while you are in the Dataflow editor, right-click an Identity Vault, then select **Add App/Driver**.

To delete an Identity Vault or driver, select it, then press the Delete key.

Adding an Application in the Dataflow Editor

1. On the toolbar, click the **Add Application** icon.
2. Browse to and select the driver set that you want this application to connect to, then click **OK**.
3. Select the driver you want to create, then click **OK**.



Designer creates a skeleton of the driver. It does not launch the Driver Configuration Wizard. If you want to configure the driver, right-click the connection icon in the Modeler, then select **Run Configuration Wizard**.

Adding Classes and Attributes

You can add classes and attributes to the dataflow.

To add a class:

1. Right-click an Identity Vault, then select **Add Classes**.
2. Select the class that you want to add, then click **OK**.

   If you want to add more than one class, press Ctrl and select the classes.

To add an attribute:

1. Right-click a class, then select **Add Attributes**.
2. Select the attribute that you want to add, then click **OK**.

   If you want to add more than one attribute, press Ctrl and select the attributes.

Adding Non-Filter Attributes

The Dataflow editor provides non-filter attributes. By default, all classes and attributes in the Dataflow editor come directly from all of the filter policies of the drivers. However, in production environments, it is common to cause data to flow a certain way directly in your Policy Script code, XSLT, or in external code that you call out to.

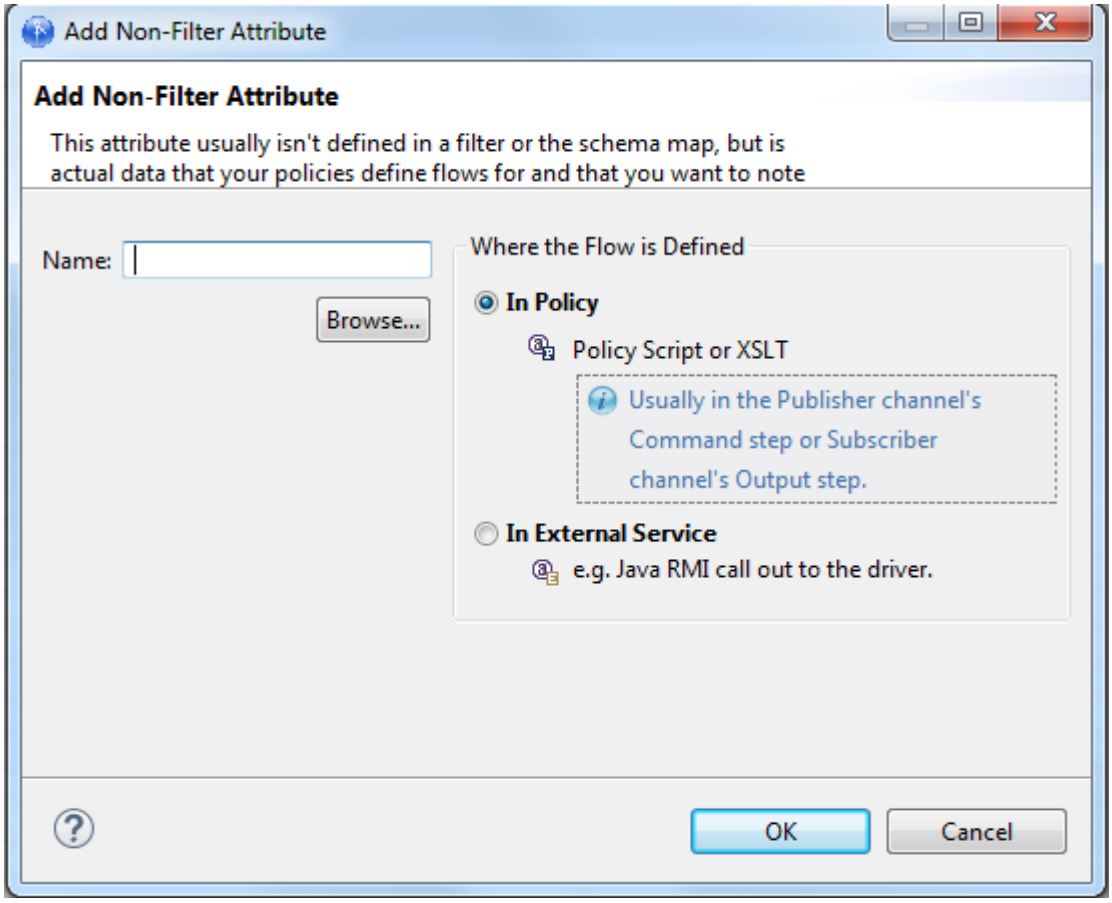
Usually, these non-filter attributes aren't defined in a policy filter (unless you're describing "augmented" processing) and aren't in the schema map. This is because they are generated outside of normal driver operations and you need them in the schema mapping rule only if the engine processes them.

Normally, non-filter attributes are operated on in the Publisher Command Transformation policy set or the Subscriber Output Transformation policy set.

The Dataflow editor lets you add the non-filter attributes to the table for documentation purposes so that you can capture the attributes and have an accurate picture of your actual enterprise dataflows.

To add a non-filter attribute:

1. Right-click the class or attribute name, then select **Add Non-Filter Attribute**.
2. Specify the name of the attribute or class, or click **Browse**, then browse to and select the attribute or class.



3 Click OK.

4 Select where the flow of the attribute or class is defined.

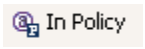
In Policy: The dataflow is defined in a policy script or an XSLT style sheet.

In External Service: The dataflow is defined in a Java RMI call to the driver.

5 Click OK.

If the non-filter attribute is defined by a policy, a small P is added to the icon. This icon distinguishes a non-filter attribute from a regular filter attribute.

Figure 8-9 *A Non-Filter Attribute*



If the attribute is defined by an external service, a small E is added to the icon.

Figure 8-10 *A Non-Filter External Attribute*



Removing Items from the Dataflow Editor

- "Removing an Identity Vault" on page 260
- "Removing Classes and Attributes" on page 260

Removing an Identity Vault

To delete an Identity Vault, select it, then press the Delete key.

Removing Classes and Attributes

To delete a class or an attribute, select the class or attribute name, then press the Delete key.

You can delete multiple objects in one Delete operation. Select the objects that you want to remove from the Dataflow editor, then press the Delete key.

Editing Items

- "Editing within the Dataflow Editor" on page 261
- "Editing Non-Filter Attributes" on page 262
- "Managing Schema" on page 262
- "Removing a Flow" on page 262
- "Changing How Data Flows" on page 262

Editing within the Dataflow Editor

As a convenience, you can edit many items within the Dataflow editor. This capability turns the Dataflow editor into a full project editor that allows you to have all the tools you need in one place. You can edit Identity Vault properties, classes, attributes, drivers, and applications.

- "Identity Vault Properties" on page 261
- "Classes and Attributes" on page 261
- "Drivers" on page 261
- "Applications" on page 261

Identity Vault Properties

Access the Identity Vault's properties pages by doing one of the following:

- Double-click the Identity Vault.
- Select the Identity Vault, then press Enter.
- Right-click the Identity Vault, then select **Properties**.

Classes and Attributes

Launch the Manage Schema tool by doing one of the following:

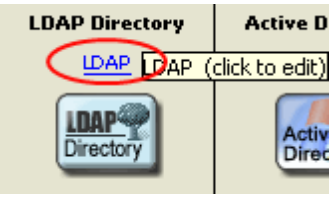
- Double-click the class or attribute.
- Select the class or attribute, then press Enter.
- Right-click the class or attribute, then select **Edit Schema**.

This tool enables you to modify classes and attributes. For more information, see Chapter 5, "Managing the Schema," on page 137.

Drivers

To access the driver's property pages, click the driver name below the application name.

Figure 8-11 *Location of a Driver Name*



Applications

Access the properties pages for the application by doing one of the following:

- Double-click the application.

- Select the application, then press Enter.
- Right-click the application, then select **Properties**.

Editing Non-Filter Attributes

The directional flow of these attributes is edited in the same way as other attributes. Right-click the arrows and select **Publish**, **Subscribe**, **Ignore**, **Reset**, or **Remove from Filter**.

Reset means that you have the value reset under certain conditions. The attribute might be in a policy filter, but in addition, you might have some manual logic that resets the value. Occasionally, resets by manual logic occur in production environments.

Managing Schema

To import, deploy, and edit the schema in the Dataflow editor, right-click an Identity Vault, then select the option that you want. All schema changes made outside of this editor are synchronized. For more information, see Chapter 5, “Managing the Schema,” on page 137.

Removing a Flow

If a particular flow (Publisher or Subscriber channel) is not defined in the policy filter’s XML, a red X replaces the Publisher or Subscriber channel arrow. This means that it’s not in the policy and there will be no flow. This scenario is essentially the same as an **Ignore Flow** icon, which is an empty white arrow. However, the distinction is useful so that you know what is actually in your policy’s XML.

To remove the flow from the XML:

1. Right-click the Publisher or Subscriber channel icon.
2. Select **Remove from Filter**.

   If a class or attribute is marked to be removed on both channels and nothing references it, Designer removes it from the Dataflow editor’s table.

Changing How Data Flows

To change the way data flows, right-click the arrow that displays the dataflow, then select the option that you want.

When you right-click the arrow that displays the dataflow for an attribute, you are presented with five options, as shown below:

- Ignore
- Notify
- Subscribe/Publish
- Reset
- Remove from Filter

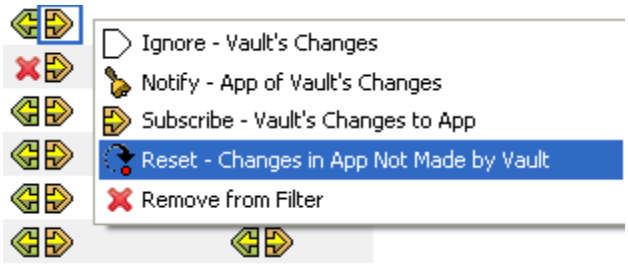
The functionality for these options changes depending on whether you have selected the left (Publisher) channel or the right (Subscriber) channel.

For the Publisher Channel:

- **Ignore - App's Changes:** Instructs the Identity Vault to ignore changes made in the application.
- **Notify - Vault of App's Changes:** Notifies the Identity Vault about changes made in the application.
- **Publish - App's Changes to Vault:** Transfers the changes made to the application into the Identity Vault.
- **Rest - Changes in Vault Not Made by App:** Resets the changes in the Identity Vault that were not made by the application.
- **Remove from Filter:** Removes the flow from the XML.

For the Subscriber Channel:

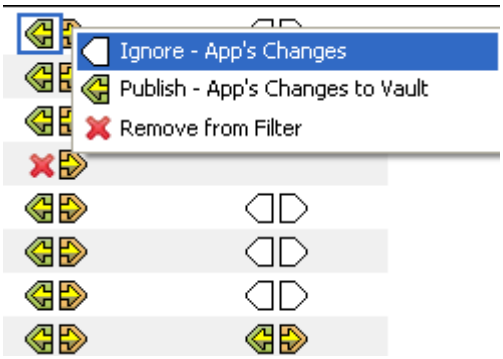
Figure 8-12 *Subscriber Channel Options*



- **Ignore - Vault's Changes:** Instructs the application to ignore changes made in the Identity Vault.
- **Notify - App of Vault's Changes:** Notifies the application about changes made in the Identity Vault.
- **Subscribe - Vault's Changes to App:** Transfers the changes made to the Identity Vault into the application.
- **Reset - Changes in the App Not Made by Vault:** Resets the changes in the application that were not made by the Identity Vault.
- **Remove from Filter:** Removes the flow from the XML.

When you right-click the arrow that displays the dataflow for a class, you are presented with three options, as shown below:

Figure 8-13 *Changing the Publisher Flow*



- Ignore

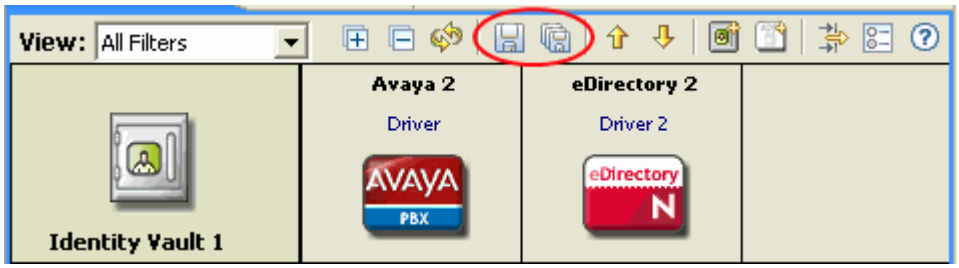
- Publish/Subscribe
- Remove from Filter

The Reset and Notify options are only available when you select an application.

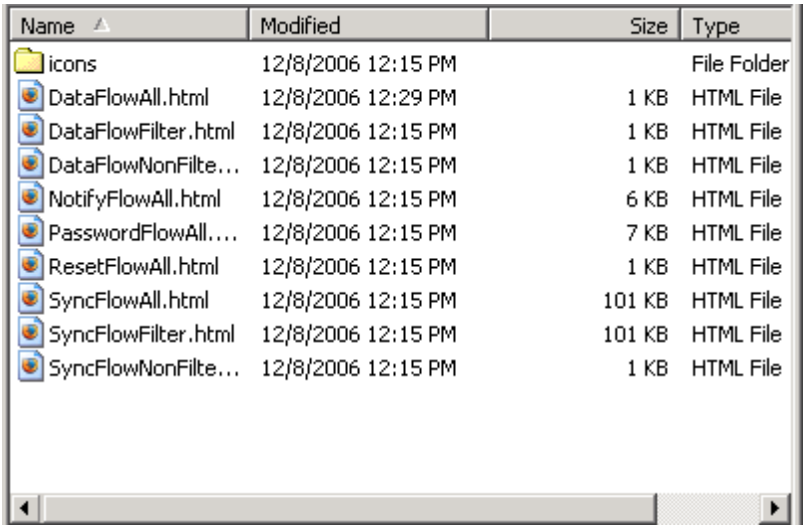
Generating HTML Reports

Designer allows you to generate HTML reports about your project.

1 Click the Save Current View to HTML icon or the Save All Views to HTML icon.



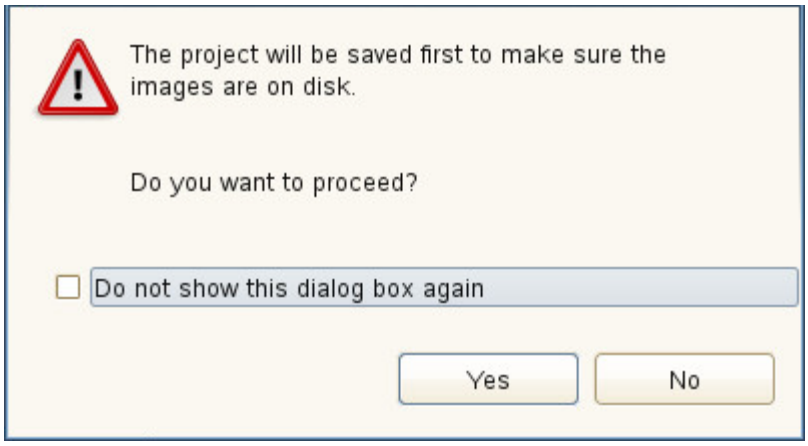
Save Current View to HTML generates a report for the current view. Save All Views to HTML generates nine reports.



Name	Modified	Size	Type
icons	12/8/2006 12:15 PM		File Folder
DataFlowAll.html	12/8/2006 12:29 PM	1 KB	HTML File
DataFlowFilter.html	12/8/2006 12:15 PM	1 KB	HTML File
DataFlowNonFilte...	12/8/2006 12:15 PM	1 KB	HTML File
NotifyFlowAll.html	12/8/2006 12:15 PM	6 KB	HTML File
PasswordFlowAll....	12/8/2006 12:15 PM	7 KB	HTML File
ResetFlowAll.html	12/8/2006 12:15 PM	1 KB	HTML File
SyncFlowAll.html	12/8/2006 12:15 PM	101 KB	HTML File
SyncFlowFilter.html	12/8/2006 12:15 PM	101 KB	HTML File
SyncFlowNonFilte...	12/8/2006 12:15 PM	1 KB	HTML File

The HTML files are automatically named. The descriptive names tell what the report is. All images that you need to support the HTML document are copied to an `icons` subdirectory where the HTML is saved. The process includes all of your custom application icons.

You are prompted to save the project to disk.



Saving is necessary to make sure that all of your icon information is in a state where it can be successfully copied.

2 Click **Yes** to save the project.

3 Browse to and select the location where you want to save the reports, then click **OK**.

The directory you select for saving is stored in Designer's memory and becomes the default directory the next time you save.

4 Click **OK** in the Information dialog box that indicates where the report is saved.

If you pin an Identity Vault and then generate a report, the report is for that Identity Vault. The Identity Vault's name is included in the HTML name.

If the Dataflow editor has multiple applications, Designer provides a scroll bar to scroll through all the applications within the Dataflow editor.

Integrating Passwords

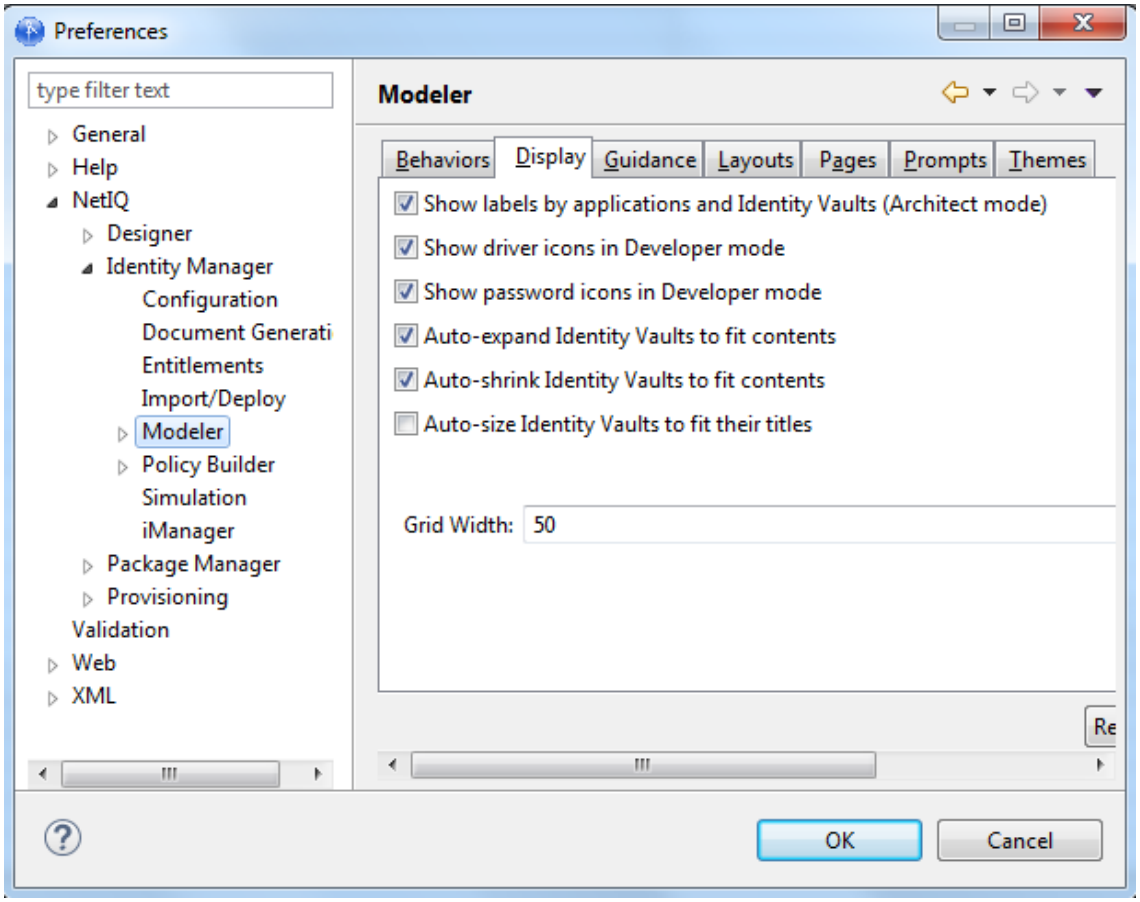
If a driver is synchronizing passwords (in at least one direction), a small password-field icon appears under the driver icon. This icon enables you to know where passwords are being synchronized.

To toggle this icon on or off:

1 Select **Window > Preferences > Identity Manager > Modeler**.

2 Click the **Display** tab.

3 Select or deselect **Show password icons in Developer mode.**



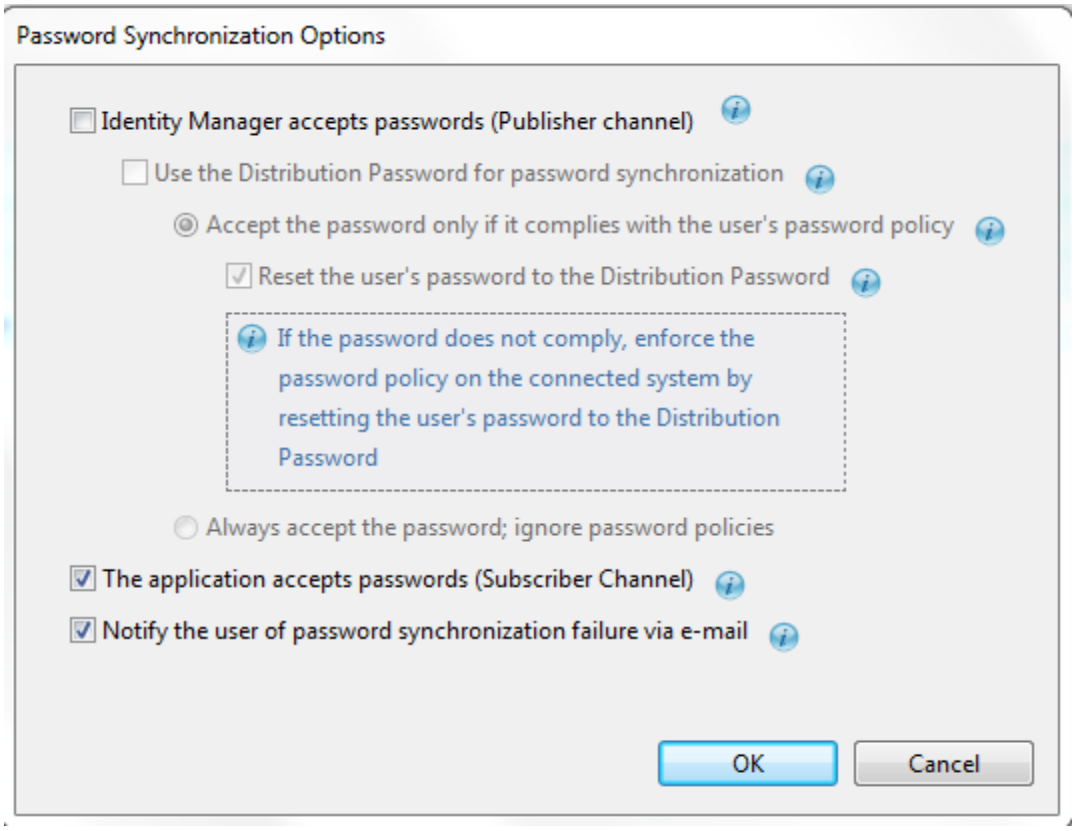
If you mouse over the password icon in Developer mode, a helpful tip explains how your passwords are flowing for each server involved in the flow.

To configure the flow of password synchronization:

1 In Dataflow mode, select **Password Sync** in the **View** drop-down box.

2 Double-click the flow arrow.

3 Select options, then click **OK**.



9 Creating and Managing Policies

Policies enable you to customize the flow of information into and out of OpenText eDirectory for a particular environment.

For example, one company might use the `inetorgperson` as the main user class, and another company might use `User`. To handle this, a policy is created that tells the OpenText Identity Manager engine what a user is called in each system. Whenever operations affecting users are passed between managed systems, OpenText Identity Manager applies the policy that makes this change.

Policies also create new objects, update attribute values, make schema transformations, define matching criteria, maintain OpenText Identity Manager associations, and many other things.

For more information about policies, refer to the following:

- *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*
- *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*
- *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*
- *OpenText Identity Manager DTD Reference (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*

10 Setting Up E-Mail Notification Templates

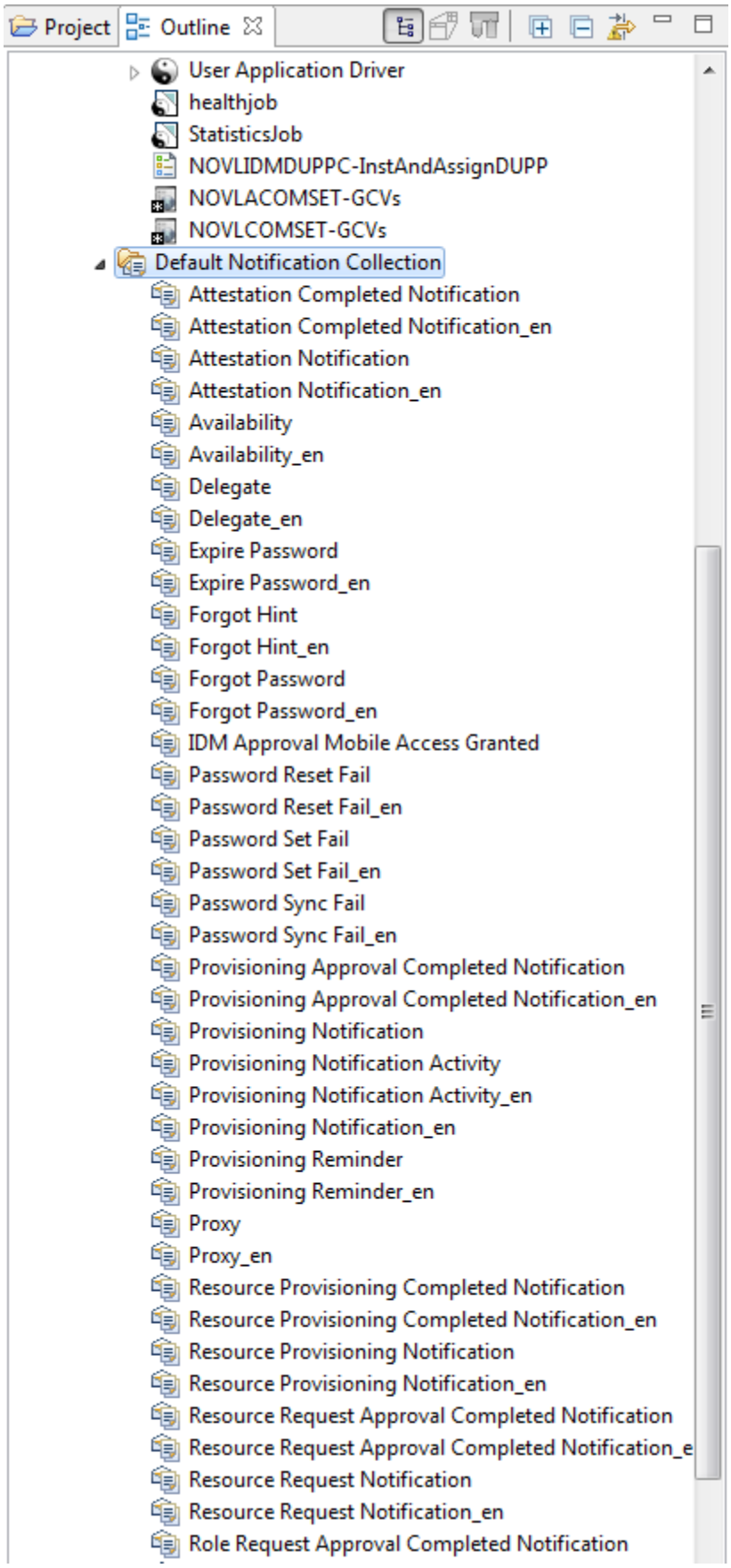
Notification templates enable you to customize and send e-mail messages that users receive when triggers occur.

- "Viewing Notification Templates" on page 271
- "Editing a Notification Template" on page 275
- "Adding and Deploying a Notification Template" on page 278
- "Policy Builder and Notification Templates" on page 280
- "Configuring the E-Mail Server" on page 280

Viewing Notification Templates

Designer provides default notification templates, which you can view or edit. To view the templates:

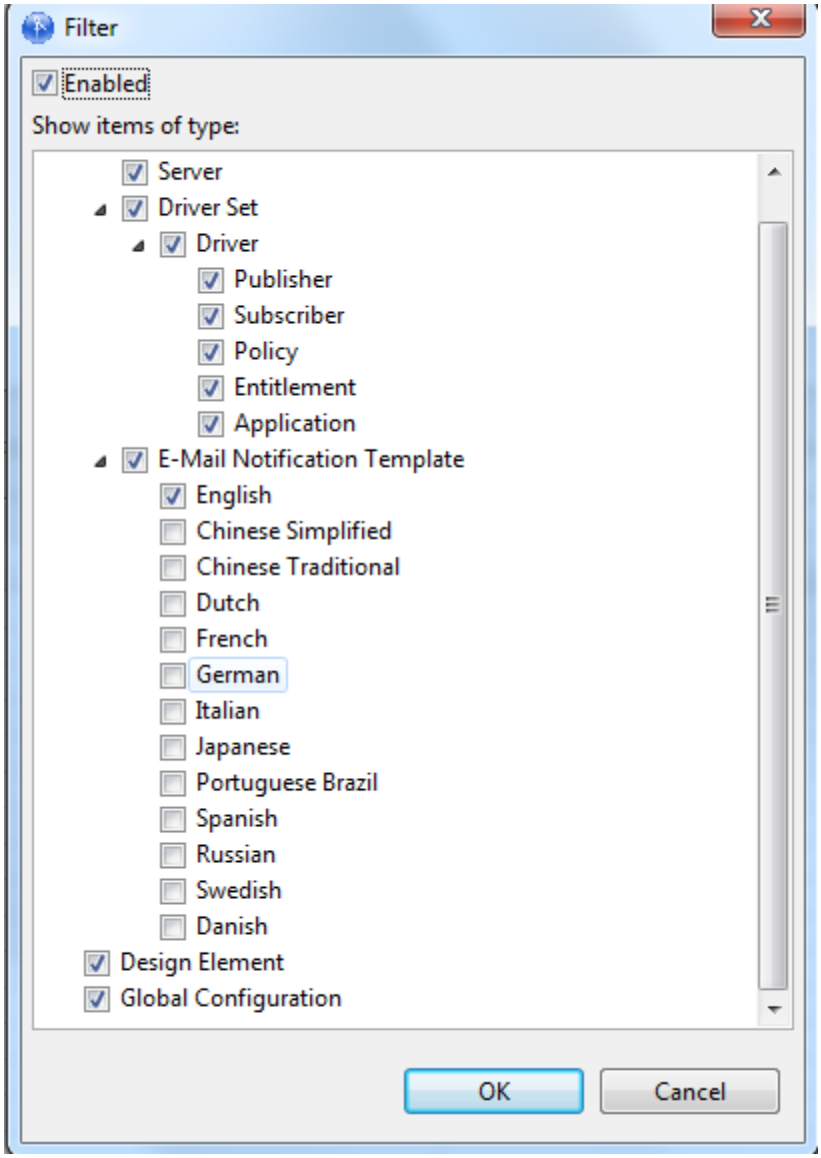
1. Select an Identity Vault in the Modeler.
2. In the Outline view, scroll to and right-click the **Default Notification Collection** for that Identity Vault.
3. Select **Add Default Templates** if you want to add the default English version of the notification templates to the Identity Vault.


Project
Outline
User Application Driver
healthjob
StatisticsJob
NOVLIDMDUPPC-InstAndAssignDUPP
NOVLACOMSET-GCVs
NOVLCOMSET-GCVs
Default Notification Collection
Attestation Completed Notification
Attestation Completed Notification_en
Attestation Notification
Attestation Notification_en
Availability
Availability_en
Delegate
Delegate_en
Expire Password
Expire Password_en
Forgot Hint
Forgot Hint_en
Forgot Password
Forgot Password_en
IDM Approval Mobile Access Granted
Password Reset Fail
Password Reset Fail_en
Password Set Fail
Password Set Fail_en
Password Sync Fail
Password Sync Fail_en
Provisioning Approval Completed Notification
Provisioning Approval Completed Notification_en
Provisioning Notification
Provisioning Notification Activity
Provisioning Notification Activity_en
Provisioning Notification_en
Provisioning Reminder
Provisioning Reminder_en
Proxy
Proxy_en
Resource Provisioning Completed Notification
Resource Provisioning Completed Notification_en
Resource Provisioning Notification
Resource Provisioning Notification_en
Resource Request Approval Completed Notification
Resource Request Approval Completed Notification_e
Resource Request Notification
Resource Request Notification_en
Role Request Approval Completed Notification

If **Default Notification Collection** is not expanded, expand it. The expanded tree shows the default notification templates. The install program no longer installs all of the notification templates with Designer.

4 Select **Add All Templates** to update all of the notification templates that are installed with Designer to the Identity Vault. You can then use the Filter option in the Outline view to filter out the notification templates that you don't want to see.

To view and edit the internationalized template files, click the **Filter** icon in the Outline view, then select languages that you want to see.



5 If you want a certain template to have all of the localized templates, right-click that template and select **Add Localized Templates**.

All of the localized templates are added for the selected template. Use the **Filter** icon to select the languages you want to see.

6 Use the templates in the Default Notification Collection to send e-mail notifications to users in the Identity Vault.

You can customize these templates with your own text. Right-click a template (for example, **Forgot Hint**), then select **Edit**.

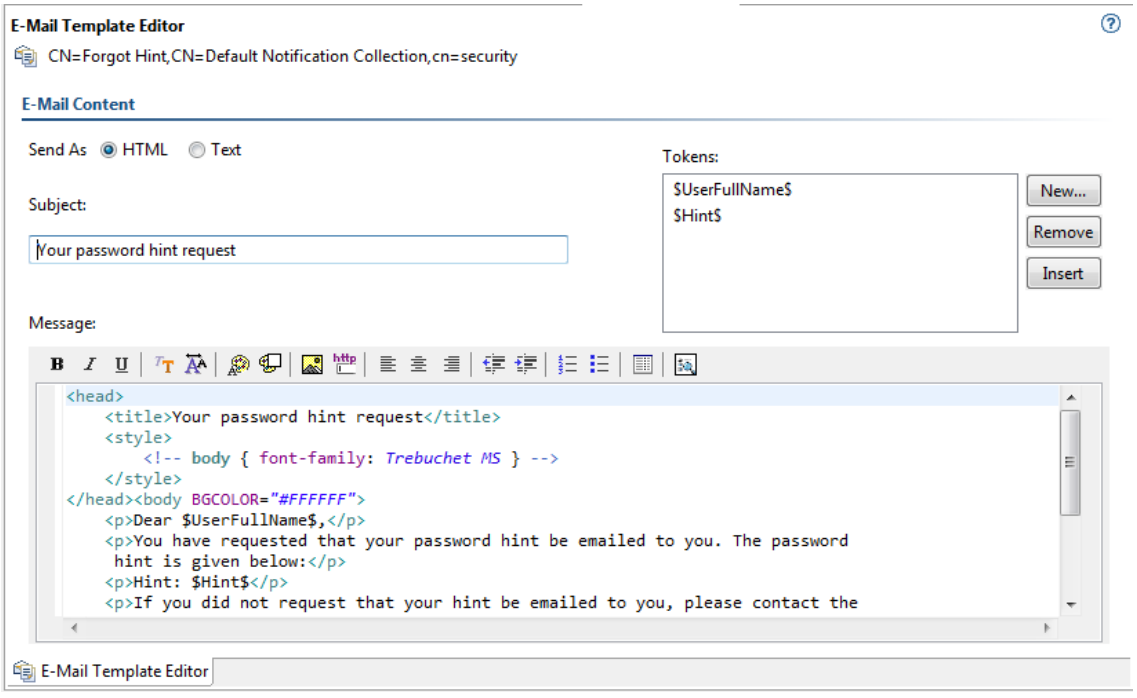
You can also open a template by double-clicking it.

Template Name	Description
Attestation Completed Notification	Sends an e-mail notification when the workflow process for your attestation request is completed.
Attestation Notification	Sends an e-mail notification when a new compliance activity is submitted that requires your attention.
Availability	Sends an e-mail notification when an availability setting has been created or modified.
Default Job Notification	Sends an e-mail notification to report results of the job as configured in the template. Contains the name of a job and any status information from the job.
Delegate	Sends an e-mail notification when a a delegate assignment has been created or modified
Forgot Hint	Sends an e-mail notification when a user forgets a password and requests a hint.
Forgot Password	Sends an e-mail notification when a user incorrectly enters a password.
Password Reset Fail	Sends an e-mail notification when a user tries to reset a password but doesn't meet password policy requirements.
Password Set Fail	Sends an e-mail notification when a user's password cannot be set in the managed system.
Password Sync Fail	Sends an e-mail notification when a user's password fails to synchronize.
Provisioning Approval Completed Notification	Sends an e-mail notification when a workflow is completed. Indicates the overall workflow and provisioning decision.
Provisioning Notification	Sends an e-mail notification to a user or manager for approval. Indicates that action is required from the user or manager.
Provisioning Notification Activity	Sends an e-mail notification to a user or manager about the activity of the provisioning notification.
Provisioning Reminder	Sends an e-mail notification when a user activity time out expires. Reminds the user or manager to act.

Template Name	Description
Proxy	Sends an e-mail notification when a proxy assignment has been created or modified.
Resource Request Approval Completed Notification	Sends an e-mail notification when a resource request has been approved.
Resource Request Notification	Sends an e-mail notification when a resource has been requested.
Role Request Approval Completed Notification	Sends an e-mail notification to a user or manager that the approval process is completed.
Role Request Notification	Sends an e-mail notification to a user or manager that a new role request requires approval.
Send Info	Sends information via an e-mail.

Editing a Notification Template

1. Select an Identity Vault.
2. In the Outline view, right-click a template (for example, **Forgot Hint**), then select **Edit**.
3. Select a format, specify a subject, add tokens, customize the message that users receive, then save and close the template.



- "Selecting a Format" on page 276
- "Specifying a Subject" on page 276

- "Working with Tokens" on page 276
- "Attaching an Image" on page 277
- "Editing a Template Message" on page 278

Selecting a Format

Select whether users receive this e-mail notification in HTML or text format.

Specifying a Subject

The subject is the text that a user views in an e-mail's Subject heading or field. You can change the text in the Subject field. You can also use tokens here. The text or tokens don't determine the name of the template.

Working with Tokens

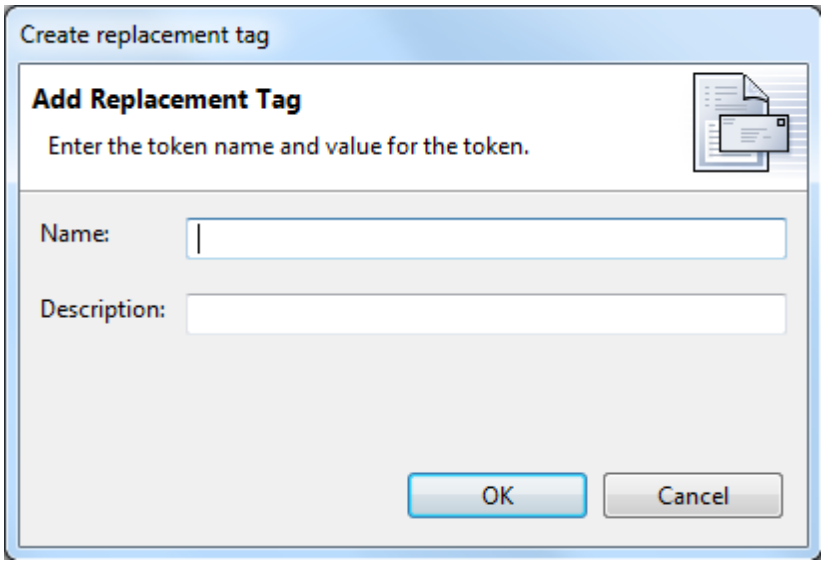
A token is a variable or replacement tag for items such as the user's name. Tokens help you personalize the message to the user.

Each template includes default tokens. For example, the Forgot Password e-mail template for sending a password to the user includes the default replacement tag named `$CurrentPassword$`.

You can define other tokens for use in the body of the message or in the subject. Your ability to do so depends on the application that uses the templates. To find out how to define additional replacement tags, see the documentation for the application. For example, OpenText Identity Manager Password Synchronization can't use a replacement tag that you create unless the policy in the driver configuration that uses the template also contains the definition of the replacement tag.

Adding a Token

1 Click **New**.



2 In the **Create a Replacement Tag** dialog box, type a name for the token.

   You don't need to type the $ characters. Designer provides them.

3 Type a description for the token.

4 Click **OK**.

When you add a token, the tag is automatically added to the XML source for the template. After you add a tag, you can edit it only in the XML Source view.

Removing a Token

To delete a token, select it, then click **Remove**.

Make sure that you don't remove tags that are needed for the body of the message.

Inserting a Token

1 In the template, click where you want to insert a token.

2 Select a token.

3 Click **Insert**.

   Designer inserts the selected token into the e-mail template.

Attaching an Image

You can attach images to the e-mail template by using the following steps:

1 Ensure that you place the image files in the correct directories depending on your platform:

   - **UNIX/Linux:** Place the images in the `/opt/novell/eDirectory/lib/dirxml/rules/manualtask/mt_files` directory.
   - **Windows:** Place the images in the `<eDirectory installation folder>\NDS\mt_files` directory.

2 In your e-mail template, use the following syntax to attach images:

```
<p> <img ALT="your image" SRC="cid:orchid.gif" height="29
width="80/> </p>
```

   where orchid is the name of the image.

   Because the file name is case sensitive, the name of the file (image) must exactly match with the file name in the directory.

3 Restart the system after placing your image files in the correct directories for your platform.

   For example, if an e-mail has already been sent, you need to restart ndsd or OpenText eDirectory for it to use the new image.

   - **UNIX/Linux:** Restart ndsd.

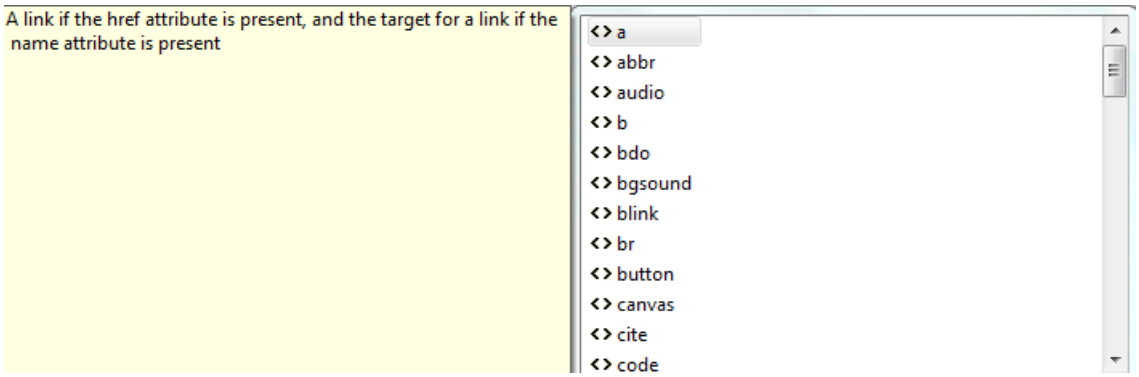
     Windows: Restart OpenText eDirectory.

4 Click **OK** to save the template.

Editing a Template Message

The text of the e-mail message appears in the **Message** field. Customize the text so that it suits your environment. Use tokens to personalize the e-mail message.

1. In the **E-Mail Template Editor**, place your cursor in the Message edit box, then press Ctrl+Spacebar.
2. Select an HTML tag by double-clicking a tag in the drop-down list.



3. Format text by using the toolbar.
4. Preview the text by clicking the Preview icon .
5. Save the template by selecting **File** > **Save**.

   You can also click the **Save** icon.

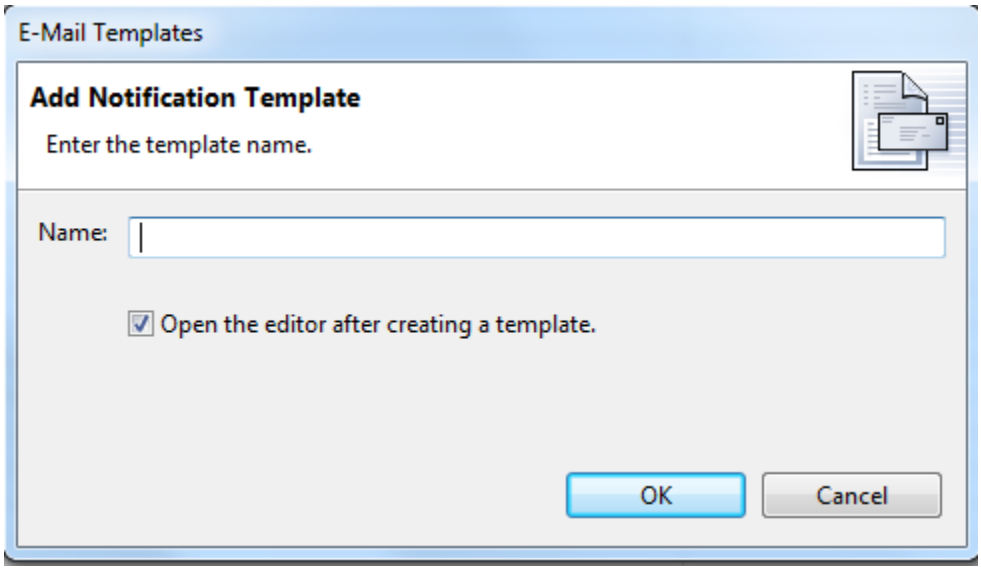
   If the code isn't valid, you can't save the template.

Adding and Deploying a Notification Template

- "Adding a Notification Template" on page 278
- "Importing a Notification Template" on page 279
- "Deploying a Notification Template" on page 280

Adding a Notification Template

1. Select an Identity Vault in the Modeler.
2. In the Outline view, scroll to **Default Notification Collection** for that Identity Vault.
3. Right-click, then select **New Template**.



4 Name the template.

5 If you want to automatically open the template editor so that you can view or edit the template, select **Open the editor after creating a template**.

6 Click **OK**.

7 Customize the text by editing the template message.

8 Click **Save** on the Designer toolbar.

Importing a Notification Template

To import a notification template from a file:

1 In the Outline view, scroll to **Default Notification Collection** for an Identity Vault.

2 Right-click, then select **Import Template from File**.

3 Browse to and select the template.

4 Customize the text for your environment by editing the template message.

To import a notification template as a live operation:

1 In the Outline view, scroll to **Default Notification Collection** for an Identity Vault.

2 Right-click, then select **Live** > **Import**.

3 Specify the host name (IP address) for the tree.

4 To authenticate, specify the user name and password.

5 Browse to and select the template, then click **OK > Continue** > **Import > OK**.

6 Customize the text for your environment by editing the template message.

Deploying a Notification Template

After you add or import a template, deploy it.

1. Right-click the template.
2. Select **Live** > **Deploy**.

Policy Builder and Notification Templates

For information about using the Policy Builder interface to send e-mail notifications, see "Send Email" and "Send Email from Template" in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies* guide.

Configuring the E-Mail Server

The e-mail server sends notification e-mails from applications that use the Notification Configuration templates.

1. Select an Identity Vault in the Modeler.
2. In the Outline view, scroll to **Default Notification Collection** for that Identity Vault.
3. Right-click, then select **Properties**.
4. Specify the host name, **From**, and authentication settings for your SMTP e-mail server.

   Host Name: The host name of your SMTP e-mail server. This can also be an IP address.

   From: When a user opens the e-mail, the text that you enter in the **From** edit box is displayed in the **From** field of the user's e-mail heading. Depending on your mail server settings, the text in this field might need to match a valid sender in the system (for example, helpdesk@company.com instead of descriptive text such as The Password Administrator). Such a match allows the mail server to do reverse lookups or authentication.

   Authenticate to the server by using credentials: Use this option for a secured SMTP server.

   If your server requires authentication before sending e-mail, specify the username and password here.

   Although the authentication information is specified here, you might also need to specify it separately for the application that is sending the notification e-mails.

   For example, Forgotten Password e-mail notifications can be sent by using the authentication information you specify here. However, notification e-mails for OpenText Identity Manager Password Synchronization require the authentication information to be provided in the driver policy that is used to send notification e-mails.
5. Click **OK**.

11 Importing into Designer

The Import feature allows you to import the following items into defined projects:

- Projects from the File System or from the Identity Vault
- Libraries
- Driver sets
- Individual drivers
- Packages
- Channels
- Policies
- Schemas

Depending on a project's complexity, importing can save you time in building and rebuilding drivers, channels, packages, and policies. For instance, after a driver, channel, package, or policy is built, you can import it into new projects and modify it to run in the new environment instead of starting from scratch on each new driver, channel, package, or policy.

You import projects, drivers, channels, schemas, and policies from an existing OpenText eDirectory tree running the OpenText Identity Manager system (Identity Vault), or from an exported project located in the file system. You import packages from the file system or the auto update feature in Designer. In Designer, use the Deploy feature to save drivers, channels, and policies into a OpenText Identity Manager server in an Identity Vault. Use the Export feature to save project, drivers, channels, and policies to a local, removable, or network directory.

What you are able to import depends where you are at within a project. To import an OpenText eDirectory object, you must have sufficient rights to access the OpenText eDirectory tree that is associated with the Identity Vault you are designing. Each of the following sections explains how to import each component of your OpenText Identity Manager solution.

During import, Designer does not import anything that is encrypted. This includes named passwords, e-mail notifications, existing certificates, and the Secure Login administrator password.

- "Importing Projects" on page 282
- "Importing a Library, a Driver Set, or a Driver from the Identity Vault" on page 292
- "Importing Packages" on page 301
- "Importing a Driver Configuration File" on page 302
- "Importing Channels, Policies, and Schema Items from the Identity Vault" on page 307
- "Using the Compare Feature When Importing" on page 315
- "Error Messages and Solutions" on page 322

Importing Projects

You can import projects from the File System or from an Identity Vault.

When you initially open the Designer utility and close the Welcome view, you have no projects that are currently available.

For information about creating a new project, see Chapter 1, “Creating a Project,” on page 19. For information about importing projects, see the following sections:

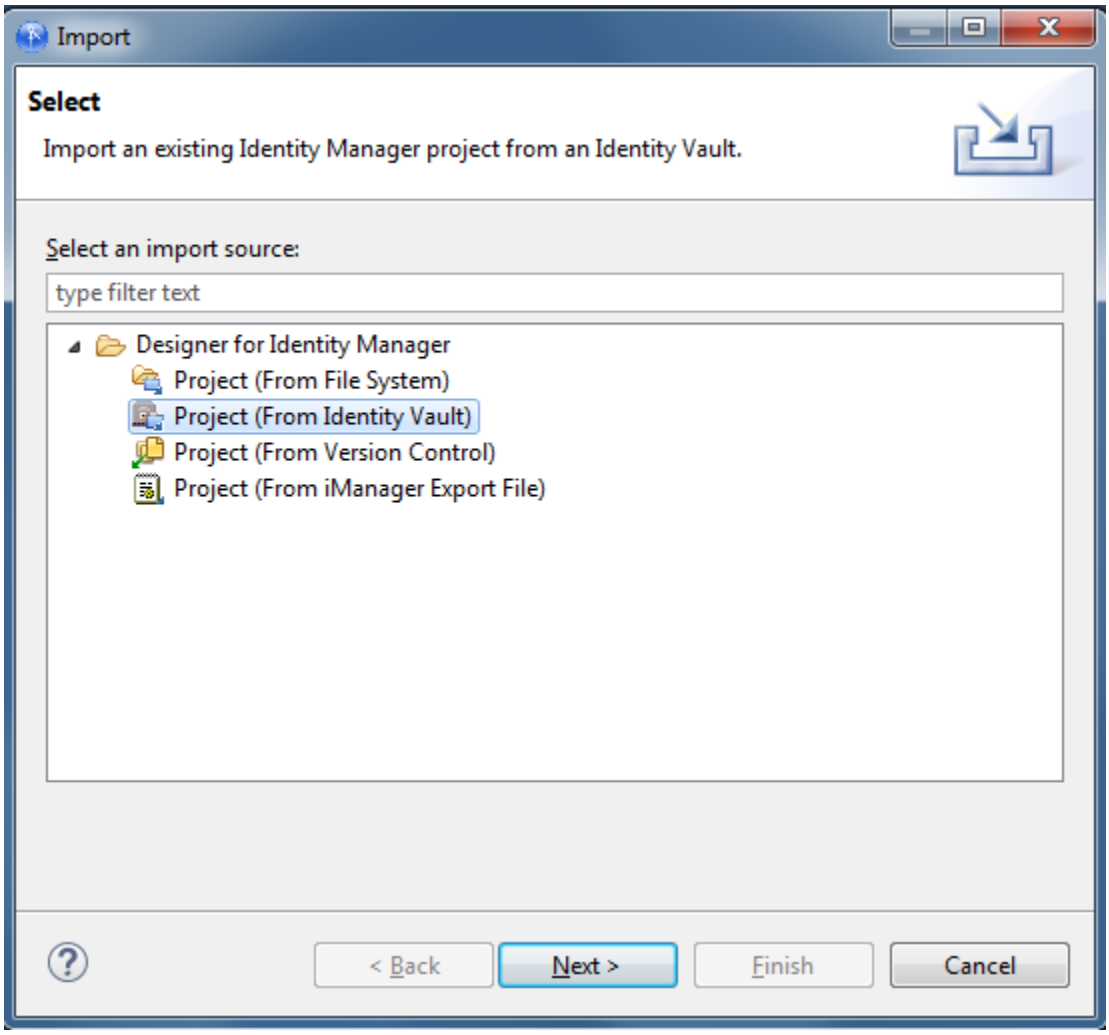
- “Importing a Project from the Identity Vault” on page 282
- “Importing a Project from the File System” on page 288
- “Importing a Project from iManager” on page 289
- “Importing a Project from a Version Control Server” on page 290

Importing a Project from the Identity Vault

1. In Designer, click **File > Import**.

   or

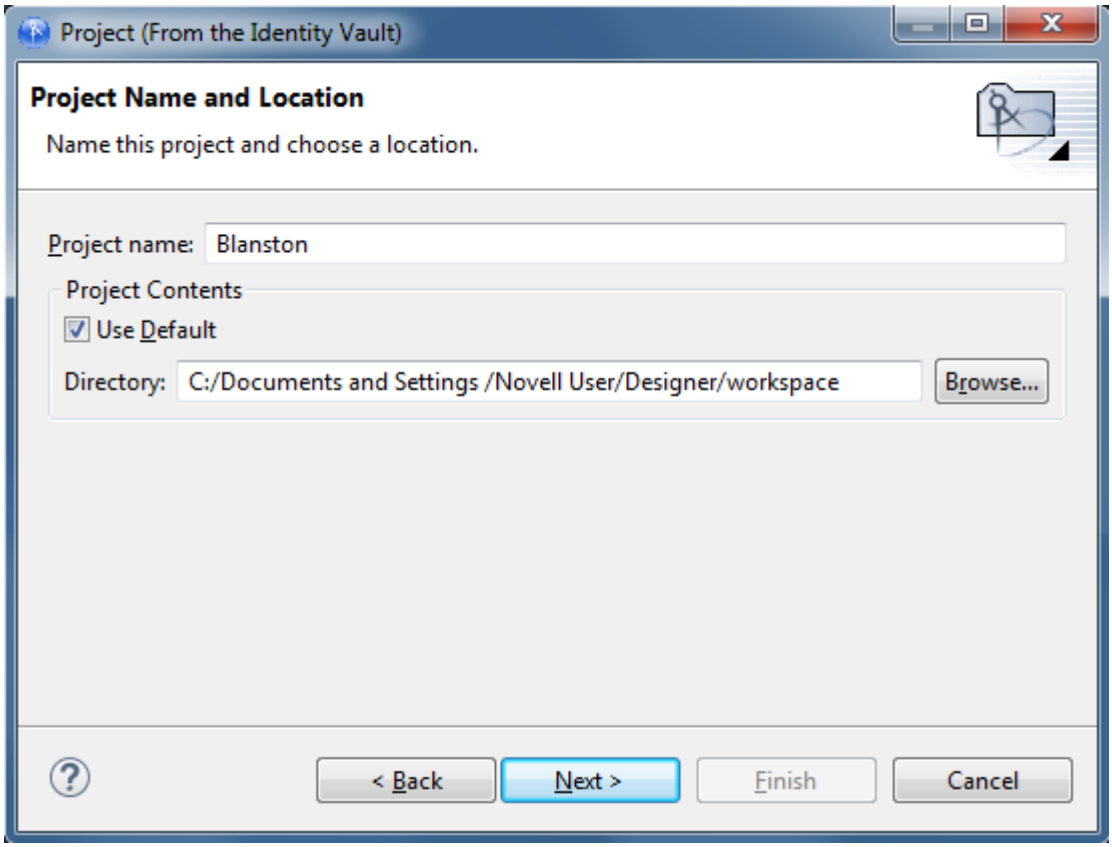
   Click **Import Project From Identity Vault** from the No Projects Available page in the Projects view, then skip to Step 3.
2. From the Import window, select **OpenText Identity Manager Project (From Identity Vault)** under the Designer for OpenText Identity Manager heading. Click **Next**.



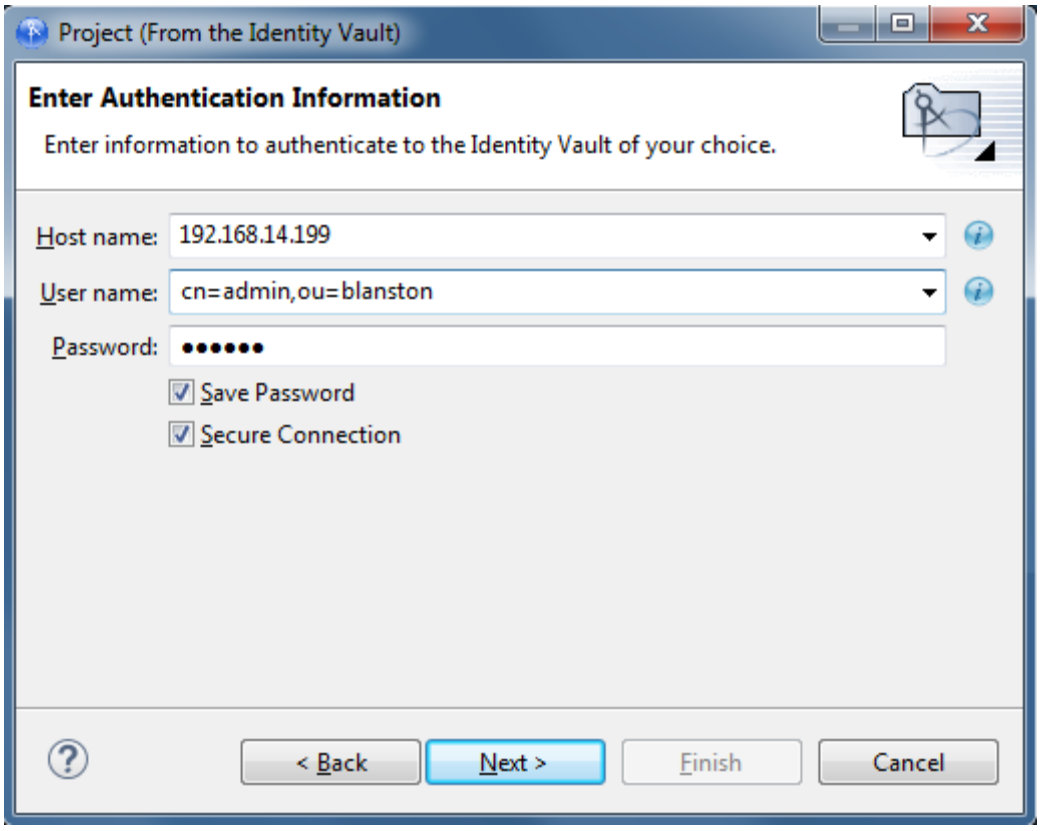
The Import window allows selections under a number of tabs, including **Designer for OpenText Identity Manager**. Selections under the **Designer for OpenText Identity Manager** tab include:

- Importing an existing OpenText Identity Manager project from the file system (the project must have a valid `.project` file).
- Importing an existing OpenText Identity Manager project from an Identity Vault.
- Importing an existing OpenText Identity Manager project from an iManager export file (Driver Set or Driver).
- Importing an existing OpenText Identity Manager project from a version control server.

3 In the Project (From Identity Vault) window, give the new project a name. Select where the contents of this project are to reside (for Windows workstations, the default is `C:\Documents and Settings\Username\designer_workspace`). Click **Next**.



4 In the Import Project From Identity Vault window, specify the information needed to authenticate to the Identity Vault (OpenText eDirectory) of your choosing. In the **Host Name** field, you can use either a tree name or the IP address of the Identity Vault.



5 Fill in the **User Name** and the **Password** fields.

You can use history drop-down lists to choose a previously entered value from a list.

6 Select **Save Password** to instruct Designer to save the password permanently. This setting will continue to keep Designer authenticated with this Identity Vault each time you import, deploy, or compare Designer objects with the Identity Vault. However, the password is saved locally in Designer's file system and is not secure.

If you do not select this option, the password is remembered only until you close Designer.

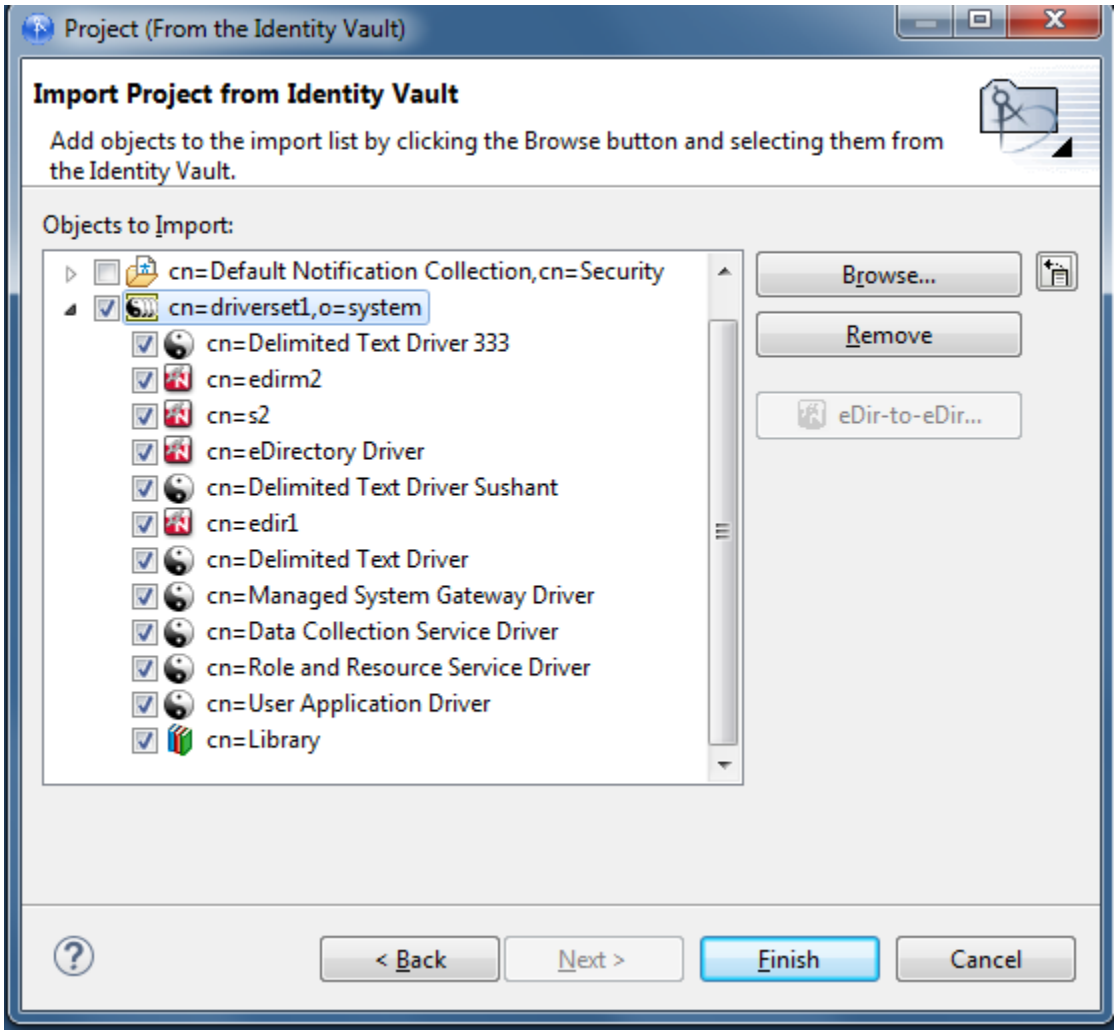
7 When you enable a **Secure Connection** between Designer and the Identity Vault, the LDAP server listens on the default secure port (636). If you configured the secure port as `700`, specify this port number in the **Host name** field. For example, `192.99.78.51:700`.

When connecting through a secure port, Designer prompts you to import the Identity Vault's Certificate Authority certificate into Designer. You must accept this certificate to establish a secure connection with the Identity Vault. For more information, see "Secure Connection" in Table 3-2 on page 71.

8 Click **Next**.

9 In the Import Project From Identity Vault page, the Identity Vault Schema and the Default Notification Collection are added as import options. If you do not want to import one of these options, select the option and click the **Remove** icon.

10 In the Import Project From Identity Vault page, click the **Browse** icon to select the object you want to import within OpenText eDirectory. Click **OK** to return to the Import Project From Identity Vault page.



11 If there are drivers you do not want to import with the driver set, select the driver and click the **Remove** icon.

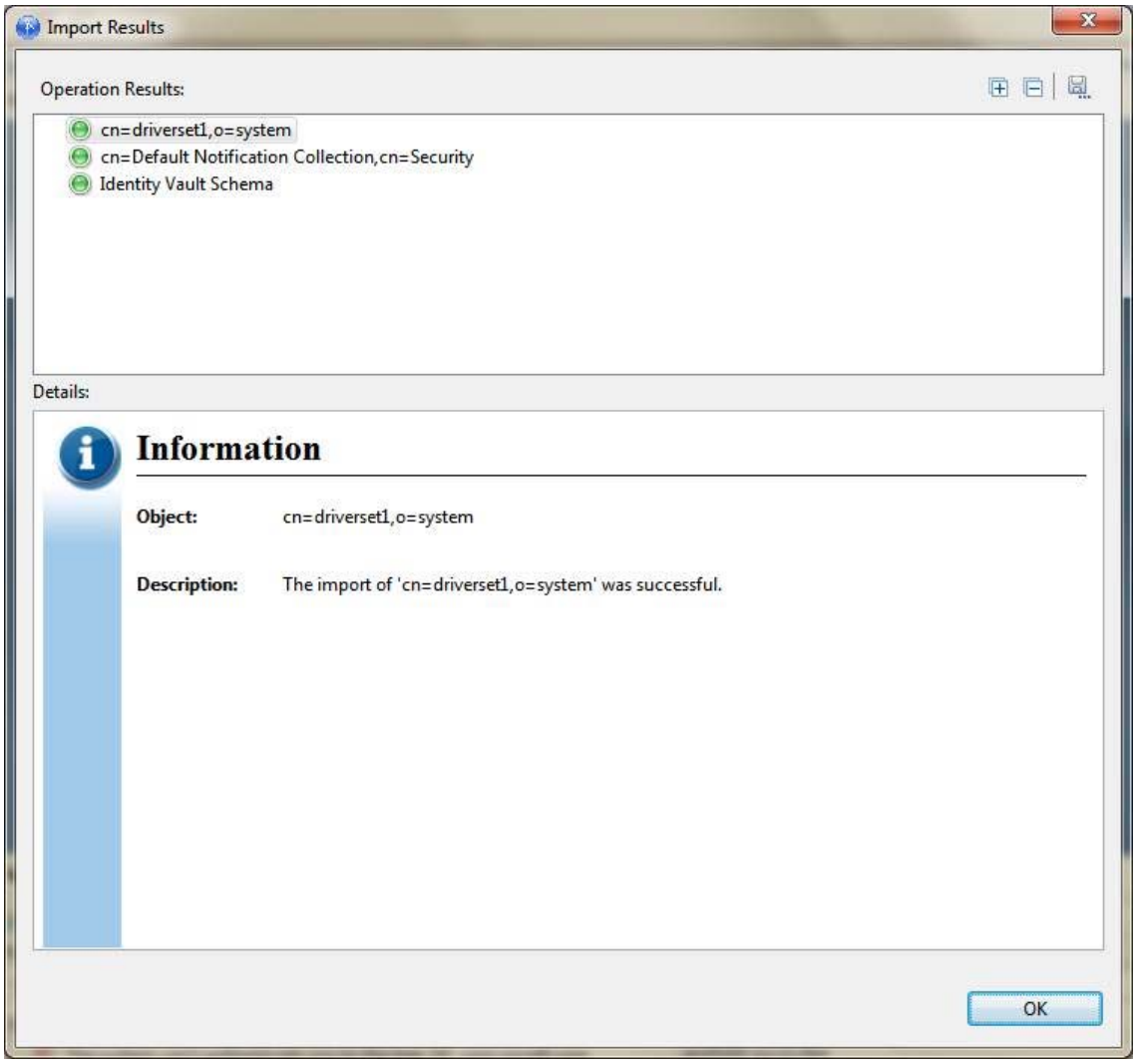
You can import multiple driver sets during the import operation. Just browse to the various objects that you want to import and add them to the list.

Driver sets that are not associated with a server have a red minus sign in the lower right portion of the driver set icon. These driver sets need a server association in order to be deployed.

12 (Conditional). You can also import policy libraries. Select the **Browse** icon to browse to and select the library you want to import, then click **OK** to add the library to the Import Project from Identity Vault page.

13 After you have selected the objects you want to import, click **Finish**.

When the driver set imports, you see the Import Results window showing you if there were any problems with the import procedure.



Errors during the import procedure are displayed with a red icon, and you see an Error description that is related to the operation results. If you have multiple errors, selecting an error displays the error's description in the **Details > Description** field. See "Error Messages and Solutions" on page 543 for further information.

14 To close the Import Results page, click **OK**.

15 (Conditional) If you are importing more than one OpenText eDirectory driver, select the OpenText eDirectory driver in the Objects to Import window and click the **eDir-to-eDir** icon in the Import Project From Identity Vault page to display the Connect to Identity Vault window, where you can import the associated driver in the other trees.

We recommend that you import both OpenText eDirectory drivers, especially if you have SSL/TLS enabled.

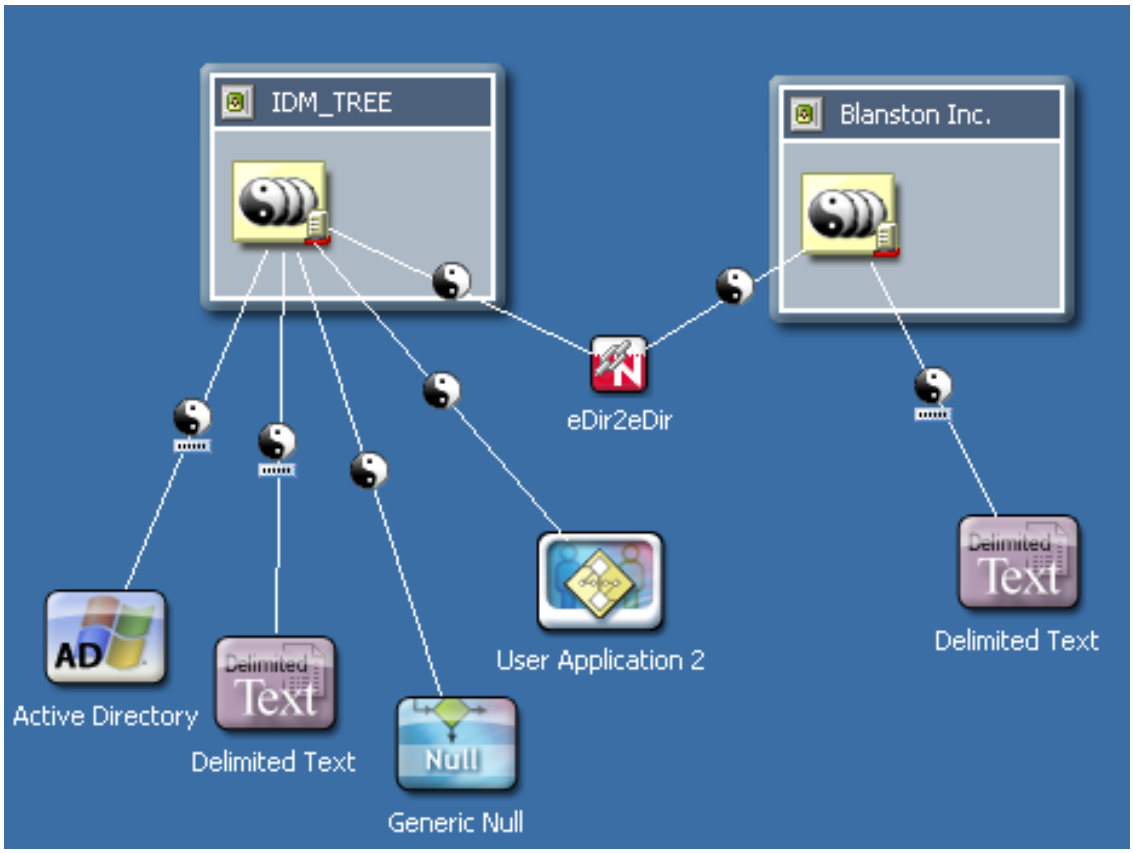
16 (Conditional) Provide the username and password, then click **Continue**. (Skip this step if you only want to import one OpenText eDirectory driver.)

17 (Conditional) If you specify the username and password and select **Continue** to import both OpenText eDirectory drivers, you then see a Browse Identity Vault window where you select the corresponding OpenText eDirectory driver. Select the driver and click **Finish**.

You are returned to the Project (From Identity Vault) window, where you can select or deselect the drivers, allowing you to choose only the drivers in a driver set that you want to import.

18 Click **Finish**.

By having both drivers available, you can view the complete data flow between the two OpenText eDirectory drivers, as well as the other drivers you selected.



Importing a Project from the File System

Earlier Designer workspaces are not compatible with Designer. Designer stores projects and configuration information in a workspace. These workspaces are not compatible from one version of Designer to another. You need to point Designer to a new workspace, and not to a workspace used by a previous version of Designer.

NOTE: While importing a project into Designer that is installed on a Linux server, ensure that there is sufficient amount of iNodes on the server. Otherwise, Designer prompts you with a message stating insufficient disk space on the server.

If you have an earlier project, you can import the project into Designer (**File > Import > Project from File System**). Be sure **Copy project into the workspace** is selected. Importing the project runs the Converter Wizard, making the project compatible with Designer architecture and placing it under your designated Designer workspace directory (`designer_workspace` by default). You must close all open editors before pointing the older workspace to Designer.

1 In Designer, click **File > Import**.

or

Click **Import from file system** from the No Projects Available page in the Project's view, then skip to Step 3.

2. From the Import window, select **Designer for OpenText Identity Manager > Project (From File System)**, then click **Next**.
3. From the Import Existing Projects page, select between the root directory or archive file. The default is **Select root directory**. Browse to the directory containing valid projects.
4. Select the directory where the exported project is saved with the `.project` extension.

   There must also be a valid project file in the selected directory, or the project does not display in the Projects window. If you have multiple projects you want to import and they are under the same directory, select the directory above the projects and click **OK**.
5. In the Import Existing Projects window, select or deselect any of the projects, then click **Finish**.
6. Make sure the **Copy Project into Workspace** option is selected to copy the contents of the project into the workstation's local workspace. (Do not use previous Designer workspaces for Designer 3.0 and above.)

   You can also import multiple projects at the same time by specifying the base or root directory where you want to start your search. The Import Wizard searches the selected directory and all of its subdirectories for valid Designer projects to import. You can then select the projects that you want to import into Designer by using the check boxes. If a project with the same name already exists in Designer, you can't import that project and you won't be able to select it from the list.
7. (Optional) If you have selected multiple projects, select whether to open these project's directories in the Model view. Designer won't open all of the projects that are imported from the file system unless you select **Open projects after imports**.
8. (Optional) You can also import projects from ZIP or TAR archives. Click the **Select archive file** selection and select the directory where the exported project is saved with the `.zip` or `.tar` extension. The whole archive is searched for valid Designer projects to import.

   If the projects you are importing need to be converted to this version of Designer, you will see the Project Converter window. See "Converting Earlier Projects" on page 417 for more information.
9. In the Import Existing Projects window, you can select or deselect any of the projects, then click **Finish**.

Importing a Project from iManager

You can create a new Designer project by selecting an iManager `.xml` export file. These include driver set and driver exports as well as OpenText sample configuration files.

1. Click **File > Import > Designer for OpenText Identity Manager > Project (From iManager Export File)**, then continue with

   or

   Click **Project (From iManager Export File)** from the No Projects Available view, then skip to Step 3.
2. Type a project name. Use the default `designer_workspace` directory for the project, or type or browse to the directory where you want to import the project. Click **Next**.

3 Select the directory where the exported project is saved with the `.xml` extension, click **Open**, then click **Finish**.

4 When you are importing a driver set or driver into a project, you are asked if you want to validate the values within the drivers you are importing. If you do not want to validate the driver configuration at this time, click **No**.

   Otherwise, click **Yes** and continue importing the project.

   You can only import the driver functionality that you saved to the `.xml` file. This file does not contain default driver configurations unless that is what you have saved.

5 Fill in the information requested in all of the Import Information Requested windows that you see for each driver in the driver set, or for each driver selected.

   The Import Information Requested windows contain different driver information from each selected driver.

6 Click **Next** or **Finish** (depending on the number of pages presented).

7 Click **OK** to close the Import Configuration Results window.

Importing a Project from a Version Control Server

The Import dialog box lists projects and enables you to select projects that you want to import. There are a number of ways to access the Import dialog box in order to import projects from a version control server, and this example covers one of those methods.

1 Select **File > Import**. or If no projects are available, select **Import from version control** from the Project view.

2 Click **Project (From Version Control)** > **Next**.

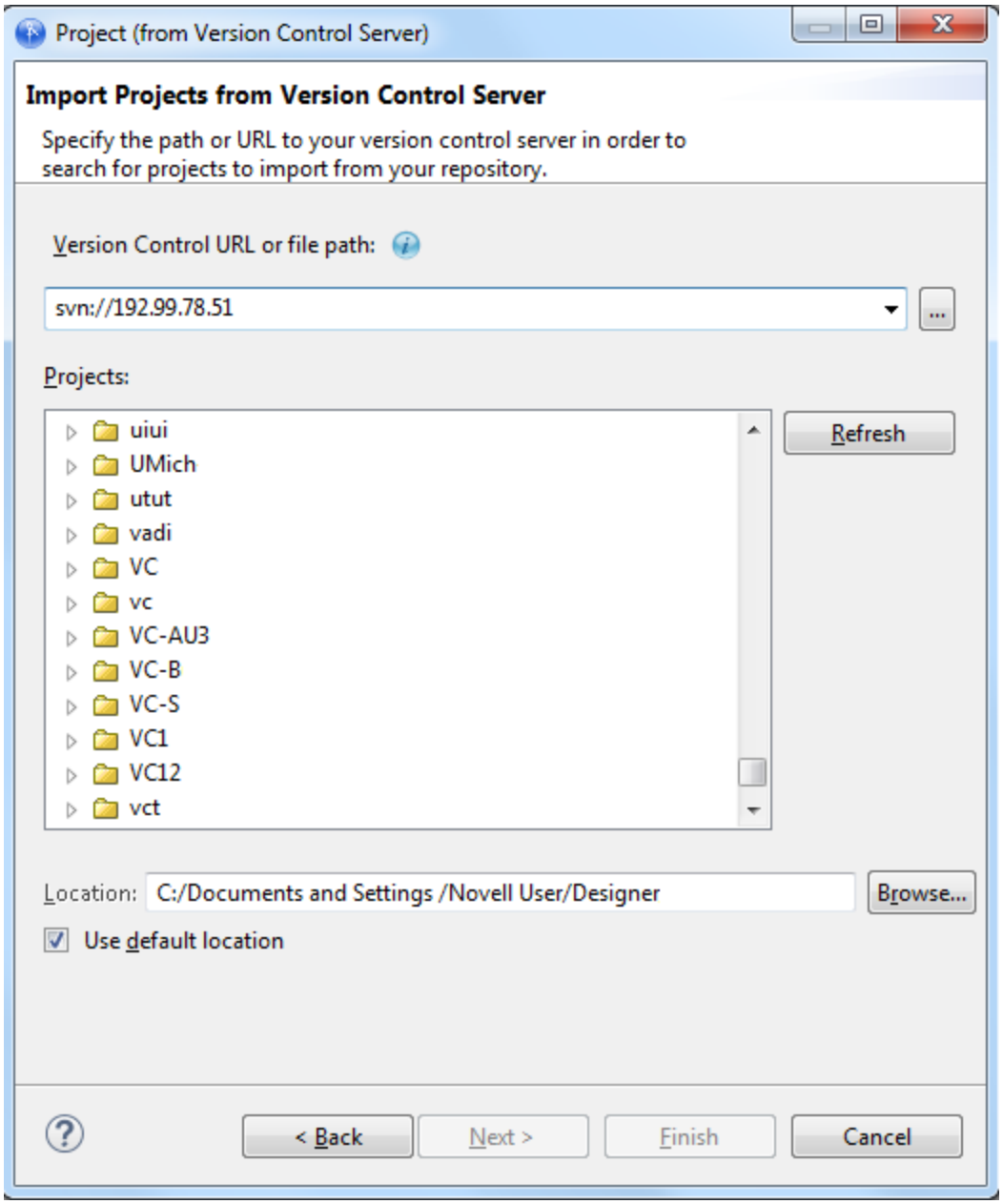
3 Type a URL in the **Version Control Server URL** field, then press Enter. For example:

   ```
   https://sun.provo.novell.com/svn
   ```

   ```
   svn://123.123.131.120/trunk
   ```

4 Provide authentication to the Subversion server if required. Depending on the type of security you have set up, you might need to supply SSH authentication, SSL client certificate authentication, or basic HTTP authentication.

5 Select a project or projects.



Version control searches for projects three levels deep from the directory specified in the **Version Control Server URL** entry.

6 Click **Finish**. At the Version Control page that shows you the version control server status, click **OK**.

The projects are imported into Designer and are added to the Project view and the Version Control view.

Importing a Library, a Driver Set, or a Driver from the Identity Vault

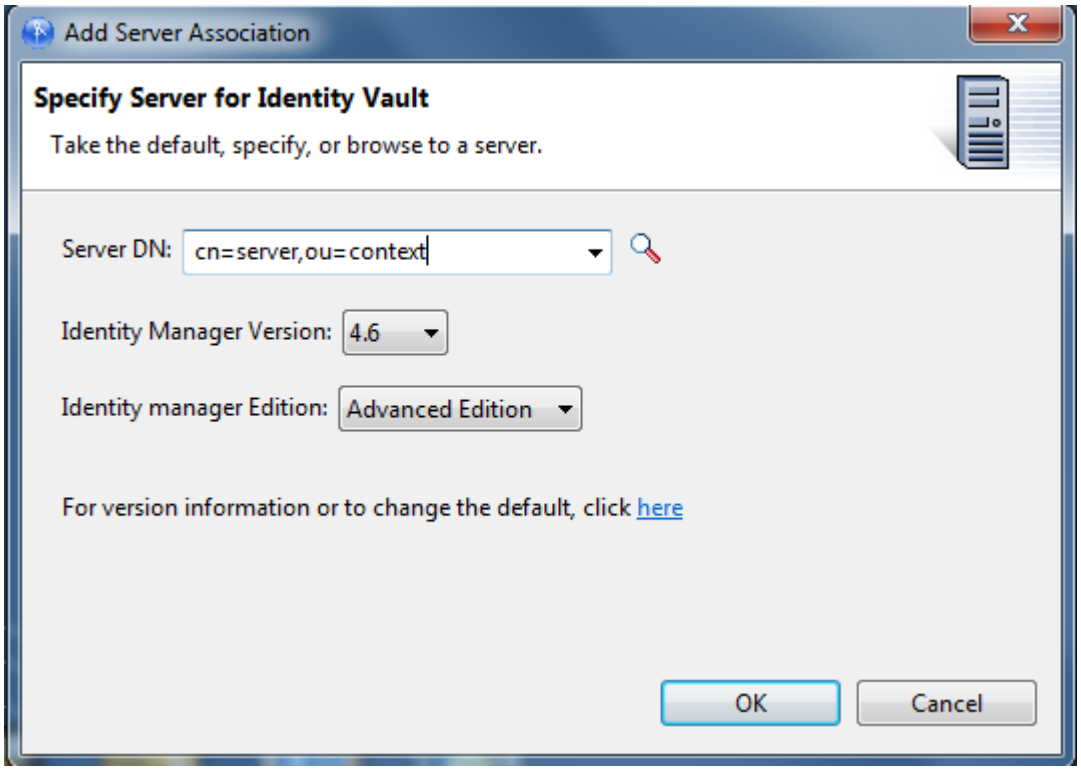
To import an OpenText eDirectory object, you must have access to the OpenText eDirectory tree that is associated with the Identity Vault.

- "Associating a Server to the Identity Vault" on page 292
- "Importing a Library from the Identity Vault" on page 293
- "Importing a Driver Set from the Identity Vault" on page 294
- "Importing a Driver from the Identity Vault" on page 298

Associating a Server to the Identity Vault

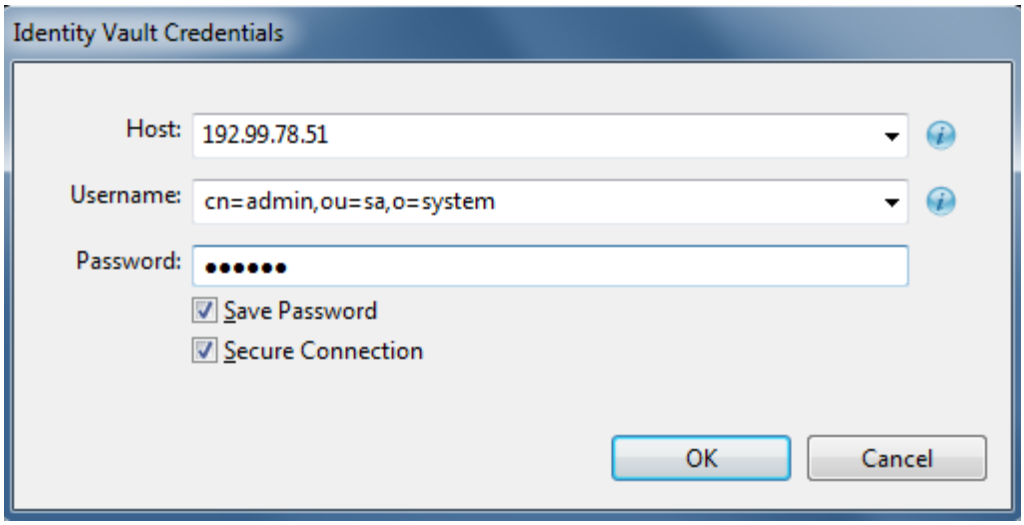
When you add a new Identity Vault to a project, you see the Add Server Association window, where you can accept the default server, specify a server, or browse to a server. The import and deploy features use the server association for later identification. To do this:

1 In the Modeler view, drag an Identity Vault icon from the Palette to the Modeler view to bring up the Add Server Association window.



2 Type the server's DN context in the **Server DN** field, or click **Browse**.

3 If you select **Browse**, fill in the appropriate host name, user name, and password in the Credentials to Identity Vault window. Click **OK**.



4 In the Browse for Server Object window, select the server you want to associate with this driver set and click **OK**.

   In the Add Server Association window, you also see the OpenText Identity Manager version displayed. This is important when importing and exporting driver sets and drivers, because you must match driver sets and drivers to the correct OpenText Identity Manager version.

5 Click the **For version information or to change the default, click here** entry for more information.

6 Click **OK** to close the Add Server Association window and add an Identity Vault to your Modeler view.

Importing a Library from the Identity Vault

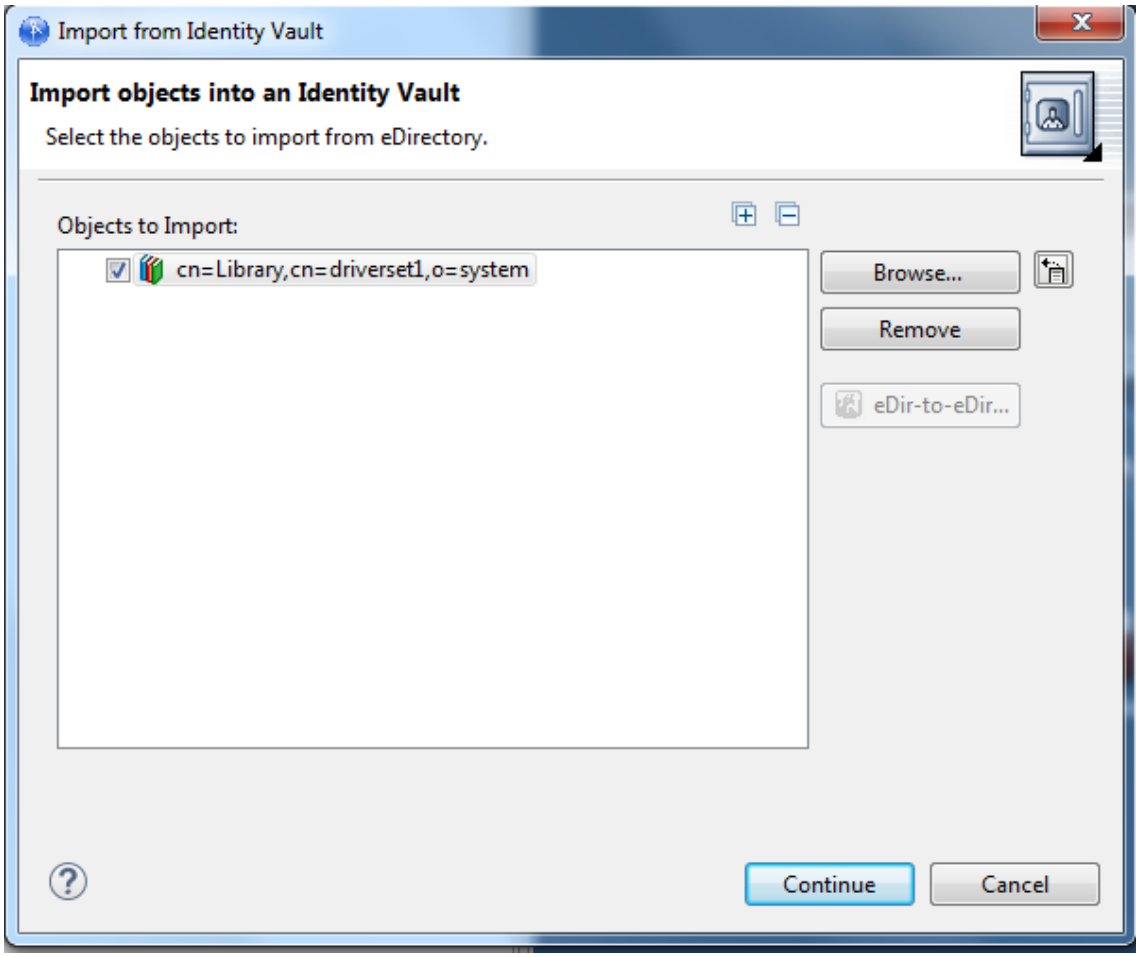
1 Right-click the Identity Vault in the Modeler view, then click **Live > Import**.

2 (Conditional) If you have not yet provided authentication information, specify it now. In the Identity Vault Credentials window, fill in the host name, the user name and password information, then click **OK**.

   Use the drop-down lists if they apply to your connection and user information. The **Save Password** option allows Designer to keep the password information for future connections to this Identity Vault. Otherwise, you will see the Identity Vault Credentials page the next time you open Designer.

3 On the Import from Identity Vault page, browse to the Library object by clicking the **Browse** icon.

4 Select the Library object and click **OK**.



The library is added to the Import from Identity Vault page.

5 Click **Continue**, then click **Import** to import the library.

6 On the Import Results page, click **OK**.

Importing a Driver Set from the Identity Vault

To import an OpenText Identity Manager Driver Set object (and all contained drivers) into an Identity Vault object in the Modeler view:

1 Right-click the Identity Vault in the Modeler view, then click **Live > Import**.

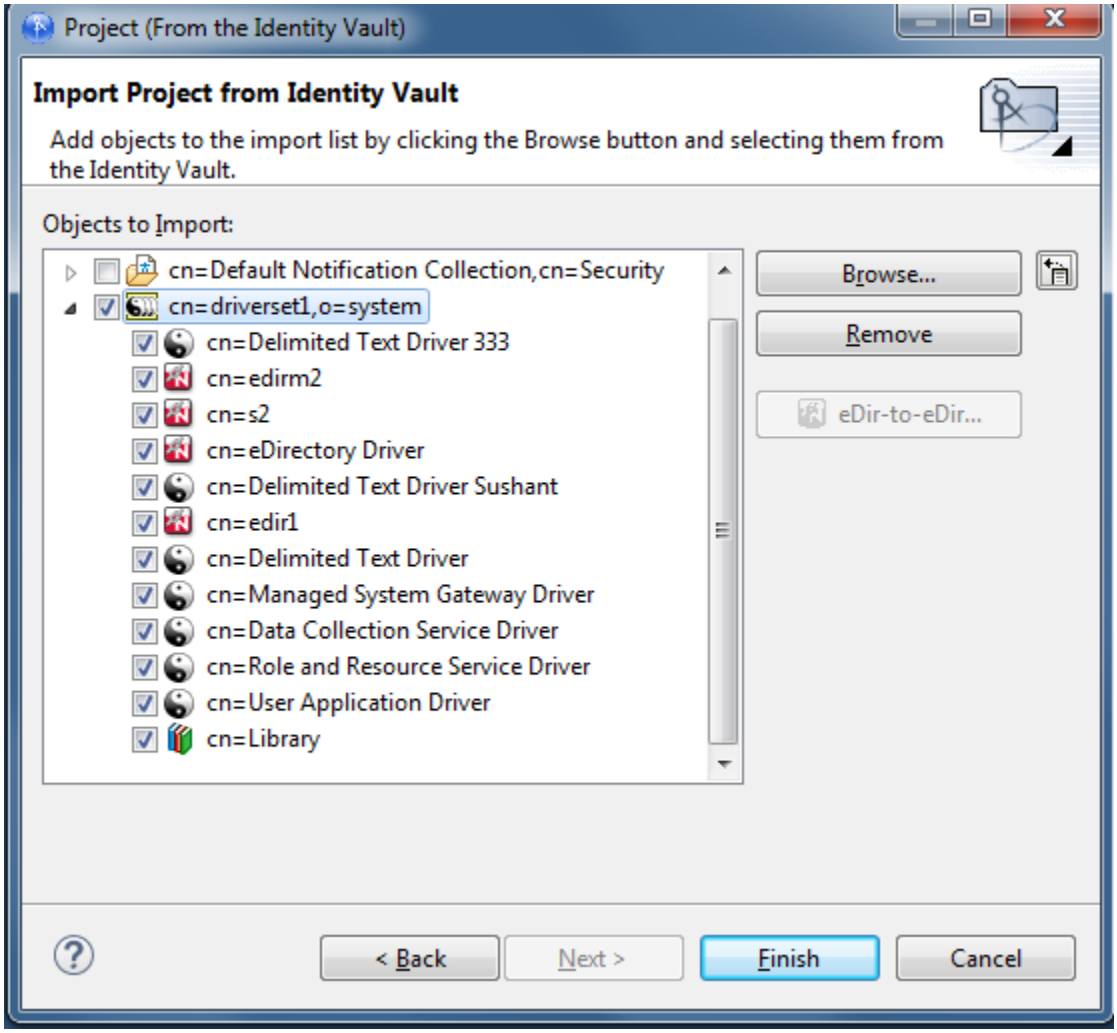
2 (Conditional) If the Driver Set that comes with the Identity Vault creation is empty, you are asked if you want to remove the default Driver Set icon from the selected Identity Vault. Click **Yes**.

3 (Conditional) If you filled out the authentication information when you initially created an Identity Vault icon in the Modeler view, go to the Properties view under the Project view. Specify authentication credentials for the selected Identity Vault, then skip to Step 5.

4 (Conditional) If you have not yet provided authentication information, specify it now. In the Identity Vault Credentials window, fill in the host name, the user name and password information, then click **OK**.

Use the drop-down lists if they apply to your connection and user information. The **Save Password** option allows Designer to keep the password information for future connections to this Identity Vault. Otherwise, you will see the Identity Vault Credentials window the next time you open Designer.

5 In the Import from Identity Vault window, browse to the driver set by clicking the **Browse** icon.



6 Select the driver set you want to import, click **OK** to place the driver set in the Objects to Import list in the Import Driver Set from OpenText eDirectory window. You can then deselect the drivers you do not want to import by deselecting the box next to the driver name. If you chose the wrong driver set, select the driver set and click **Remove**. Otherwise, click **Continue**.

You can import multiple driver sets during the import operation. Just browse to the various objects that you want to import and add them to the **Objects to Import** list.

Driver sets that are not associated with a server have a red minus in the lower right portion of the driver set icon. These driver sets need a server association in order to be deployed. An error displays if the application can't authenticate to the OpenText eDirectory tree you have selected.

7 (Conditional) If you are importing one or more OpenText eDirectory drivers, select the driver in the Objects to Import window, then click the **eDir-to-eDir** icon.

8 (Conditional.) If you fill in the user name and password and click **Continue** to import both OpenText eDirectory drivers, you then see a Browse Identity Vault window where you select the corresponding OpenText eDirectory driver. Select the driver and click **OK**.

   You are returned to the Import Driver Sets from OpenText eDirectory window, where you can select or deselect the drivers, allowing you to choose only the drivers in a driver set that you want to import.

9 Click **Continue**.

   This brings up the Import Summary window, where you can see all of the Driver Set objects that are being imported into Designer. This summary uses the same format as the Compare window (see "Using the Compare Feature When Importing" on page 315 for further information).

10 Click **Import** to continue.

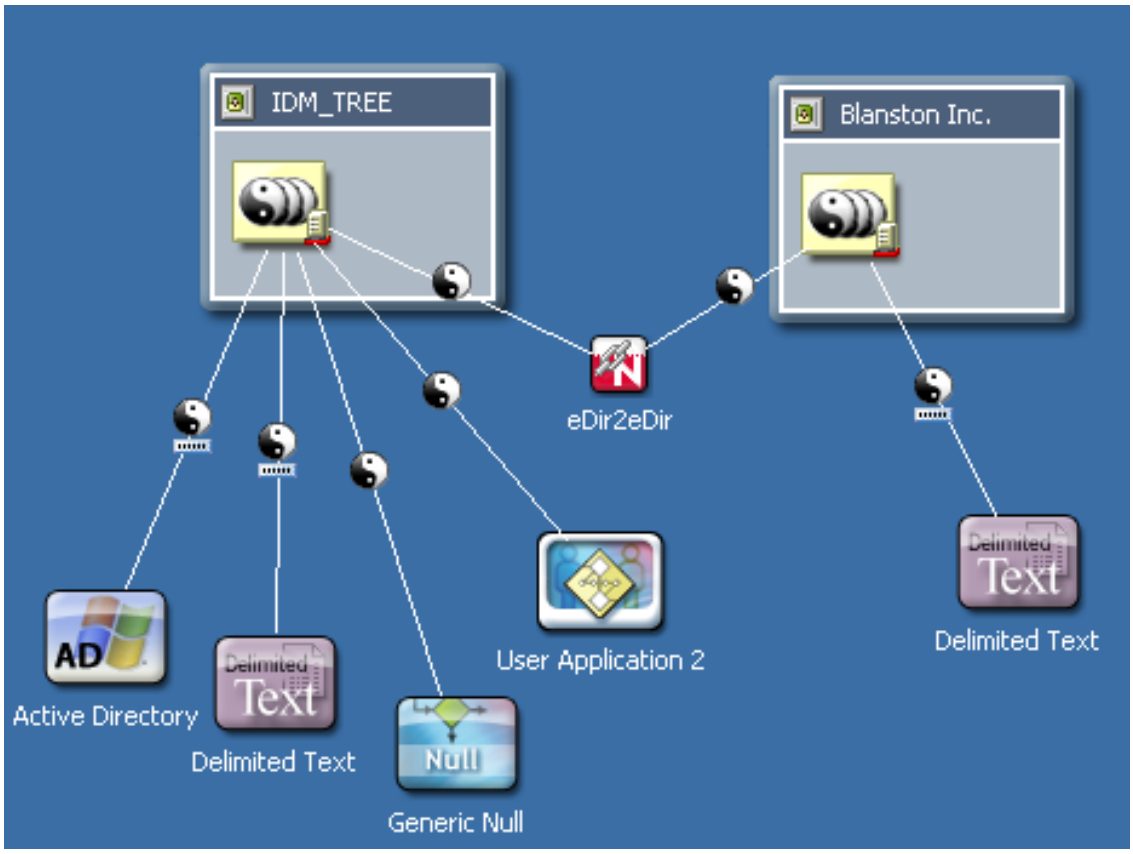
11 (Optional) As the import operation progresses, you are asked to associate a server with the Identity Vault. Select the option that best suits your needs.

   - **Allow default server to be created:** Creates a dummy server with global configuration values and other elements that are associated with this project until you specify an actual server for the project. Make sure you have designated a correct Identity Vault server when you deploy the driver set.
   - **Specify a server:** Brings up the credentials screen, allowing you to designate a host server, a user name, and password for the Identity Vault server for this project.
   - **Don't create a server now:** Skips all associations for this project. You need to fill in the host information before you deploy this driver set.
   - **Remember selection - don't prompt again:** Continues to use whatever server option you choose the next time Designer needs to associate a server with an Identity Vault.

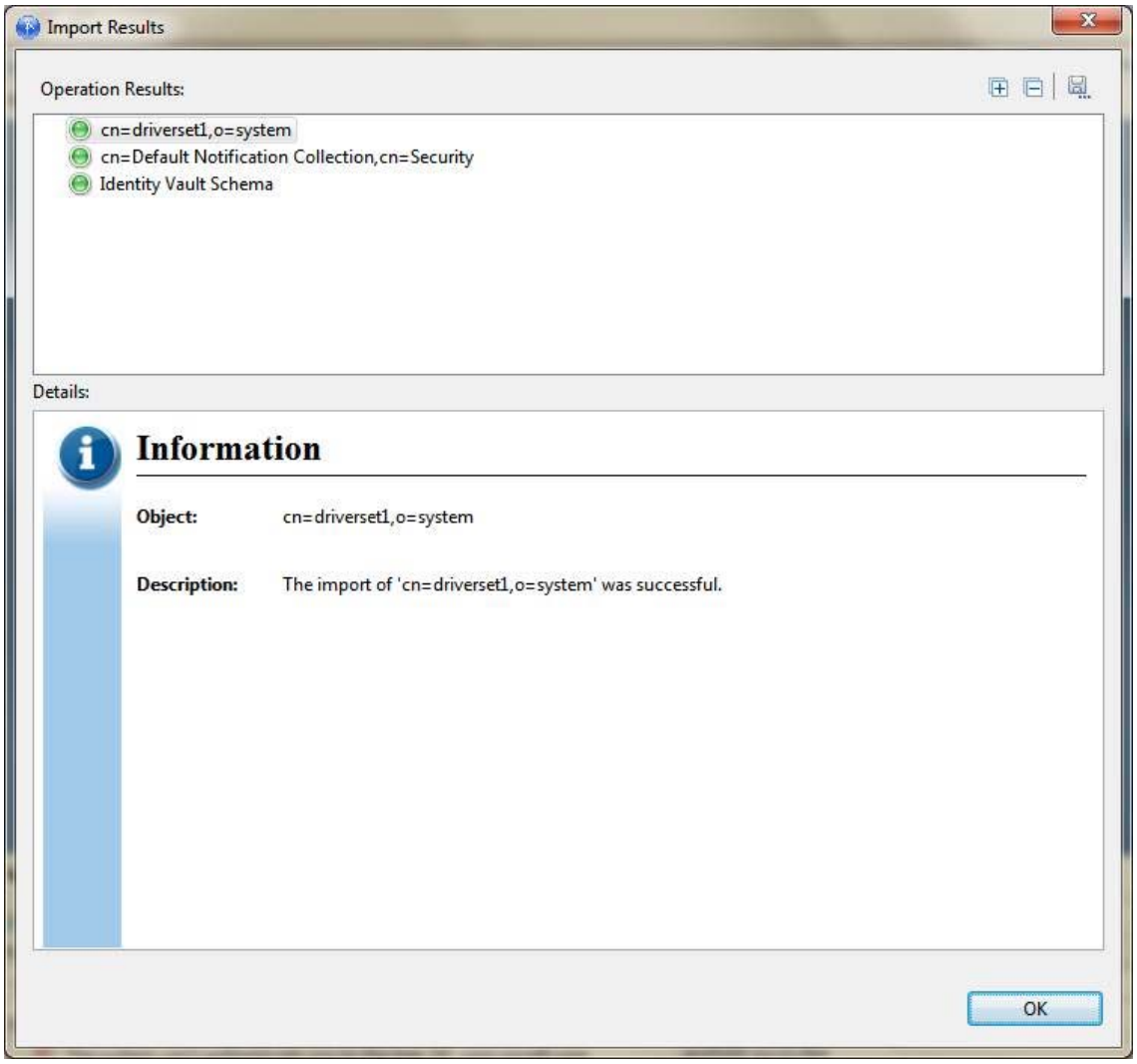
12 After you decide your plan of action and select the option you want, click **OK** to continue the import procedure.

13 Click **Finish**.

   If you selected in Step 7 to connect OpenText eDirectory drivers, you can view the complete data flow between the two drivers, as well as the other drivers you selected.



When the driver set imports, you see the Import Results window, showing you if there were any problems with the import procedure.



Errors during the import procedure are displayed with a red icon, and you see an Error description that is related to the operation results. If you have multiple errors, selecting an error displays the error's description in the **Details > Description** field. See "Error Messages and Solutions" on page 543 for further information.

14 Click **OK** to finish the import process.

Importing a Driver from the Identity Vault

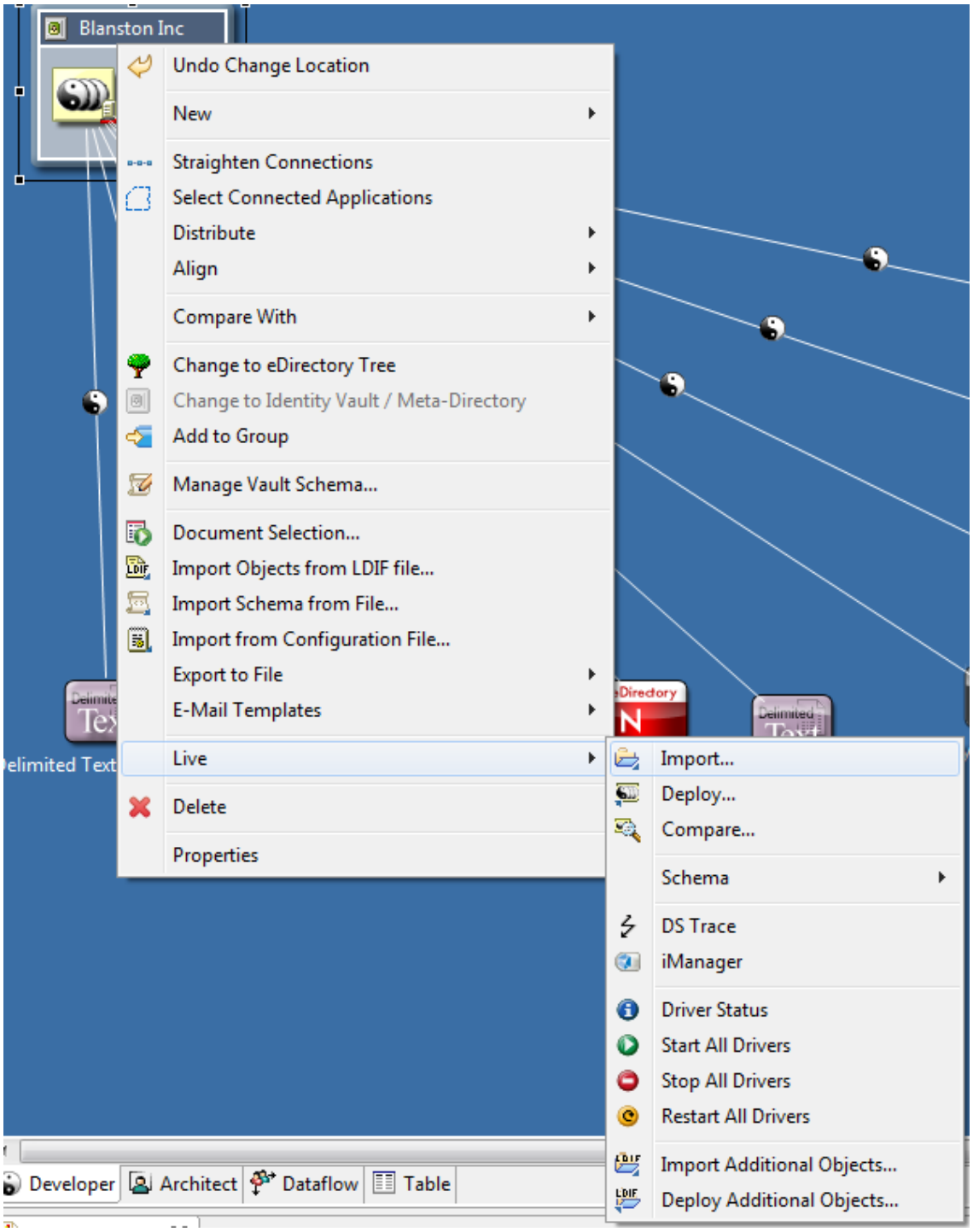
To import an OpenText Identity Manager Driver object (and all contained channels and policies) into a driver set:

1 Select an Identity Vault in the Modeler view.

If you have added a new Identity Vault to a project, see "Associating a Server to the Identity Vault" on page 292 first, then return to Step 2.

2 Verify that the authentication credentials in the Properties view for the Identity Vault are correct.

3 Right-click a Driver Set object within the Identity Vault, then select **Live** > **Import**.



4 (Conditional) If the Identity Vault is not authenticated to the OpenText eDirectory tree, you see the Identity Vault Credentials window asking for the hostname, username and password. Provide the information, then click **Next**.

5 In the Import from Identity Vault window, click **Browse** to select a Driver object from the Identity Vault.

6 Click **OK** to place the driver in the Import from Identity Vault window, then click **Continue** to install the driver and bring up the Import Configuration window.

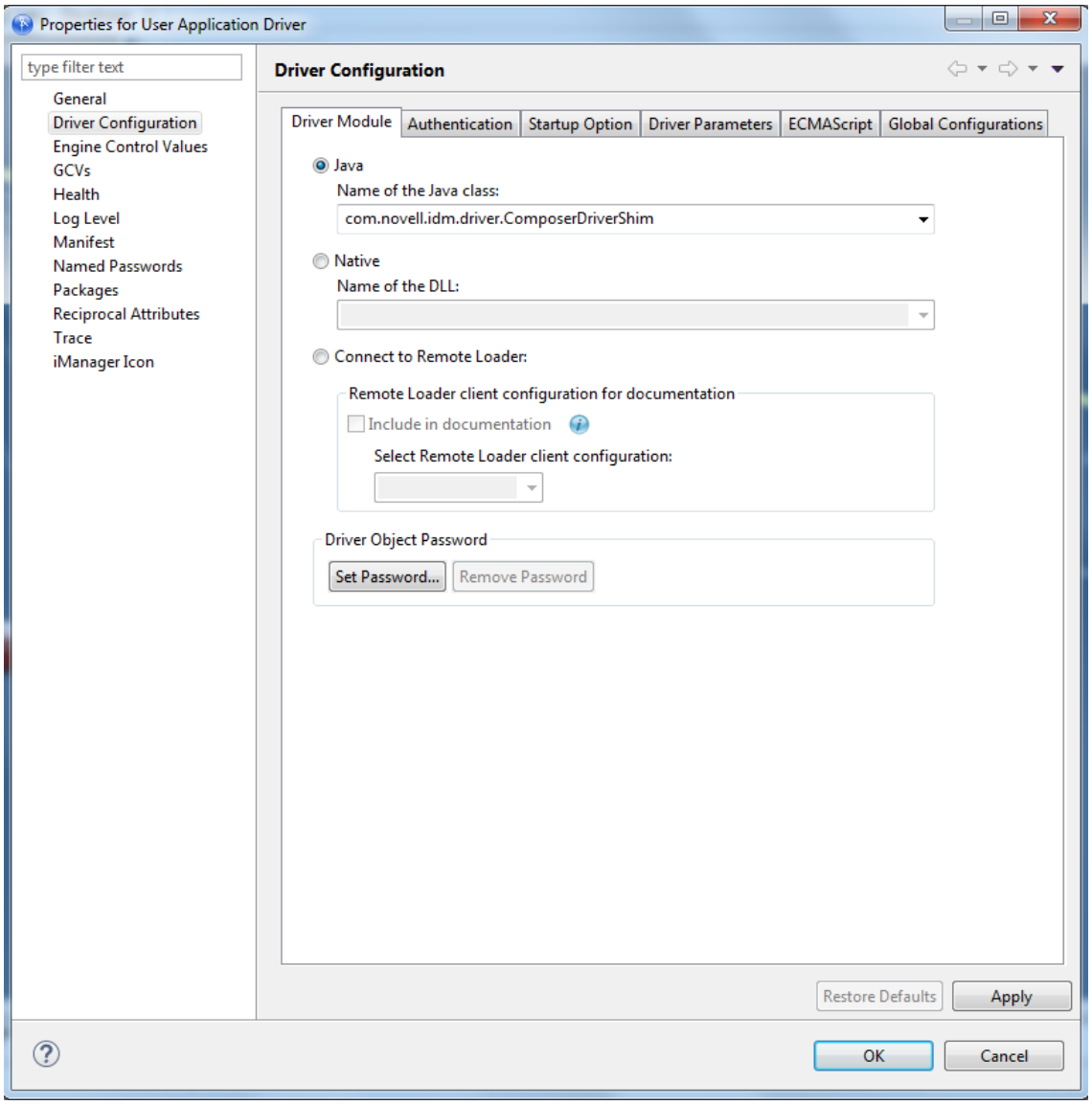
7 In the Import Configuration window, select **Configure** to edit the driver configuration, or select **Close** to close the Import Configuration window.

Most drivers cannot run with default values. You need to modify the driver configuration values and parameters so the drivers can work properly in your network environment.

You also see the Import Configuration window when you drag an application from the Palette to a driver set in the Modeler view.

When you select **Configure**, the driver's Property page with the **Driver Configuration** option is displayed.

8 Fill in the required values and parameters that are necessary to have the driver run in your network environment.



The two required options for every driver are **Driver Configuration** and **GCVs**. However, because each driver contains different values and parameters, you need to consult the driver manual for specific values. Go to the OpenText Identity Manager Drivers Guide Website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/), then select the manual for the driver you are configuring.

9 (Conditional) If you are importing one or more OpenText eDirectory drivers, we recommend that you connect to those eDirectory drivers during the import process. Select the eDirectory driver in the Objects to Import window, then click the **eDir-to-eDir** icon.

10 (Conditional) Fill in the user name and password for the other eDirectory tree and select **Continue** to import both OpenText eDirectory drivers.

11 (Conditional) In the Browse Identity Vault window, select the corresponding eDirectory driver, then select the driver and click **OK**.

You are returned to the Import Drivers from OpenText eDirectory window.

When the driver imports, you see the Import Results window showing you if there were any problems with the import procedure.

Errors during the import procedure are displayed with a red icon, and you see an Error description that is related to the operation results. If you have multiple errors, selecting an error displays the error's description in the **Details > Description** field. See "Error Messages and Solutions" on page 543 for further information.

12 Click **OK** to finish the import process.

Importing Packages

In Designer 4.0 and later, packages replace driver configuration files. You can still import driver configuration files, but from this point on, new content is delivered in packages.

Designer is the only tool that allows you to manage packages. iManager can detect if a driver is created with packages. However, if you make changes to the driver in iManager, Designer cannot track these changes. If you install an updated package, there is a possibility your changes can be overwritten. It is a best practice to always make driver configuration changes through Designer and not iManager.

Packages must be imported into the package catalog, then the packages are installed on the Identity Vault, driver sets, or drivers. The package catalog is an object that is only displayed in Designer and it holds all of the packages that are installed into a Designer project.

To import packages into the package catalog:

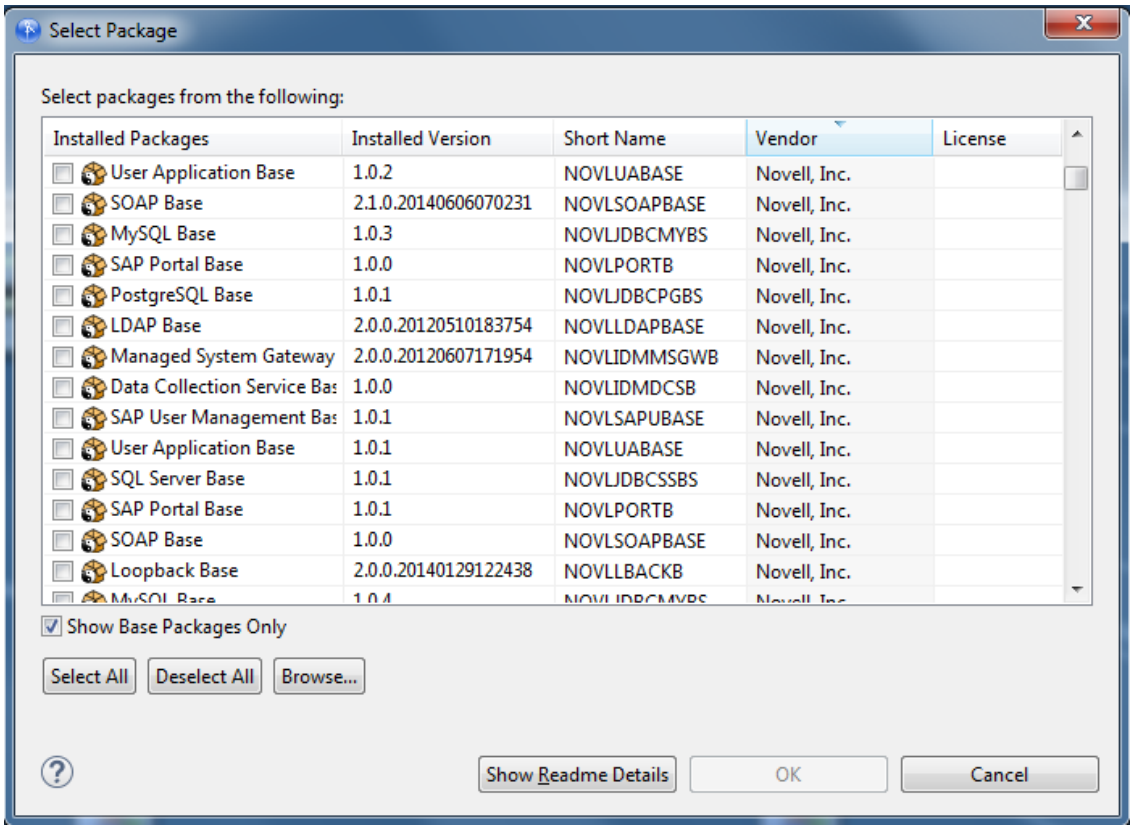
1 Select the package catalog object in the Outline view, then right-click and select **Import Package**.

2 Select a package from the list.

or

Click **Browse**, then browse to and select a package on the file system.

If all of the available packages are imported, the list is empty.



3 Click **OK** to import the package.

After the package is imported, you must install the package into the Designer project on an Identity Vault, driver set or driver. To install a package, see "Installing Packages" on page 175.

Importing a Driver Configuration File

In Designer 4.0 and later, packages replace driver configuration files; however, you can still import driver configuration files. Any new functionality for the drivers is contained in packages, not in the configuration files.

You can import an OpenText Identity Manager driver configuration file into the selected driver set for a project by using the **Import from Configuration File** option, which imports an XML configuration file that can be a driver set, an individual driver, driver channels, or policies. If you import a driver configuration file that contains only a policy, Designer creates the framework for parent containment objects, such as a channel, a driver, or a driver set. Such parent containment objects do not contain attributes; they are only the framework of the channel, driver, or driver set from where the policy came.

You can import a configuration from a file in three ways:

- "Importing an OpenText Identity Manager Project from the File System" on page 303
- "Importing a Driver Configuration from a File in the Modeler View" on page 303
- "Importing from a File through the Outline View" on page 305

Importing an OpenText Identity Manager Project from the File System

The **Import an OpenText Identity Manager Project from File System** option allows you to import an OpenText Identity Manager project that has been saved to the file system through the **Export** > **File System** option. The project must have a valid `.project` file and accompanying file structure for the project to completely import. For information about importing a project, see "Importing a Project from the File System" on page 288.

Importing a Driver Configuration from a File in the Modeler View

You can import a previously exported configuration file or one of the sample `.xml` driver configurations that are included with Designer.

To import a configuration file into a driver set:

1. Select an Identity Vault in the Modeler view.
2. Right-click a Driver Set object within the Identity Vault, then select **Import from Configuration File**.
3. In the Import a Driver Configuration File window, type the name of the configuration file, or browse to and select the `.xml` file.

   If you use the Browse feature, by default Designer takes you to the following directories:

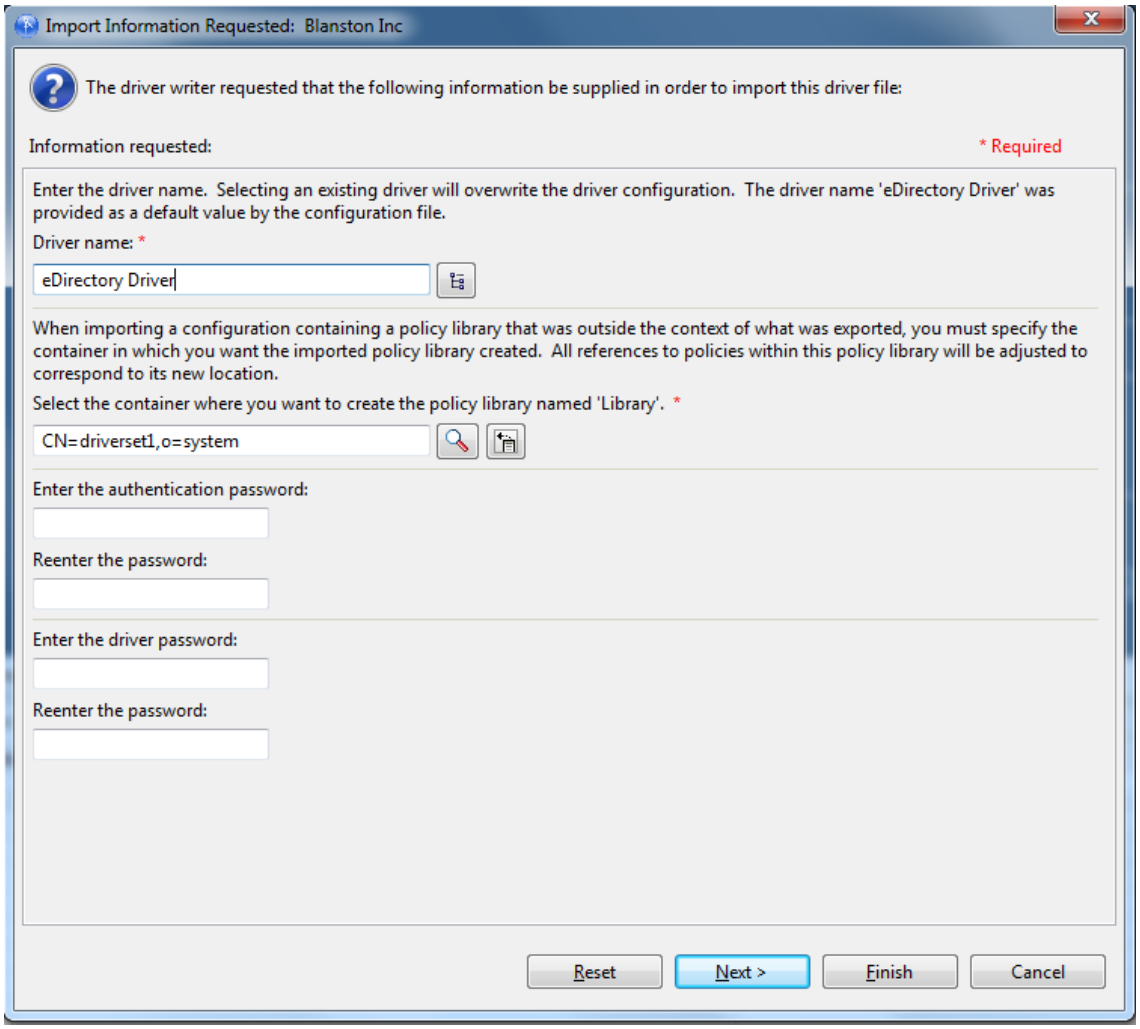
   - For Windows:

     ```
     C:\Program
     Files\Novell\Designer\eclipse\plugins\com.novell.idm_<version><time
     stamp>\defs\driver_configs\current\drivername
     ```

   - For Linux:

     ```
     /home/username/designer/eclipse/plugins/
     com.novell.idm_<version><time stamp>/defs/driver_configs/current/
     drivername
     ```

   You can use one of the `.xml` configuration files in a selected directory or you can browse to a directory containing an exported configuration file.
4. Click **OK**.
5. Complete the import by filling out the Import Information Requested prompts for the driver configuration file.



- The template varies, depending on the driver configuration file selected and the state in which the file was saved. Saved files might only prompt for a driver name, but need other values in order to work in a network environment.

  The pre-Identity Manager 3.6 Driver Configuration Wizard adds different policies to the driver, depending on which options you select when you initially import the driver. You cannot change an option that you did not initially choose, because the information is not included in the driver. You must delete the driver and create a new one through the Driver Configuration Wizard.

  WARNING: Do not use the Driver Configuration Wizard on the `.xml` file that you are importing. The Wizard brings up the Import Information Request forms, but these forms are pulled from the default driver that come with Designer and will overwrite the driver you are importing. Use this method only if you need to start over.

- The Identity Manager 3.6 Driver Configuration Wizard adds all policies when the driver is imported, and are not lost if you did not select an option in the Import Information Request forms. If this is a driver configuration file that came with Identity Manager 3.6, you can change the driver's values through the Properties page.

  If the driver needs other values and parameters in addition to what appears on the Import Information Requested template, close the template, right-click the driver line in the Modeler view and select **Properties > Driver Configuration** and **GCVs** to fill in what you need. You might also need to fill in GCVs at the driver set level.

Because each driver contains different values and parameters, consult the driver manual that matches the installed driver at the OpenText Identity Manager Drivers Website (https://www.netiq.com/documentation/identity-manager-4.10-drivers/). Then select the manual for the driver you are configuring.

6 Click **OK**, then click **OK** in the Import Configuration Results window.

7 You might have imported a single driver or a collection of drivers (a driver set). If you are importing a driver set, repeat Step 4 through Step 5 for each driver in the driver set.

8 When you are finished with each imported driver, click **OK** at the Import Configuration Results window.

Importing from a File through the Outline View

You can use the Outline View to import driver sets, drivers, channels, and policy configuration files from the file system. The following example demonstrates how to import a driver, but the procedure also works for the other files.

1 Double-click the **System Model** icon under a project name in the Project view. This brings up the project model in the Modeler view.

2 Click the **Outline** tab.

3 Right-click the Driver Set object and select **Import from Configuration File**.

4 In the Import a Driver Configuration File window, type the name of the configuration file, or browse to and select the `.xml` file.

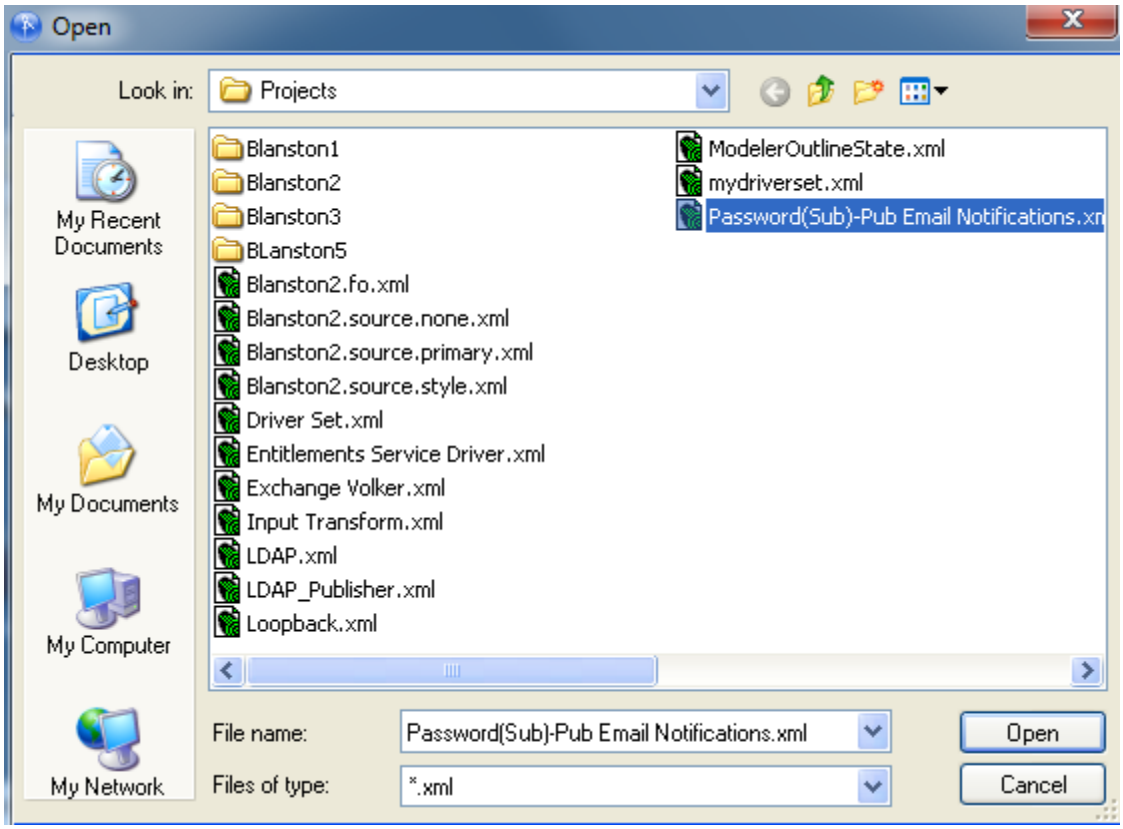
  When a driver is exported, Designer uses the name of the driver set or driver object, to which you can add dates. In this example, it is an Active Directory driver that was exported June 26th and is now being imported.

5 Click **Open**, then click **OK** to import the configuration file.

To import a policy into a driver set:

1 In the Outline view, click the **Driver Set** icon, then click **Import from Configuration File**.

2 In the Import a Driver Configuration File window, browse to or specify the XML configuration filename.



3 Click **Open**, then click **Save** to import the selected policy.

Each policy is saved to its own .xml file. By default, Designer uses the name of the policy or rule selected.

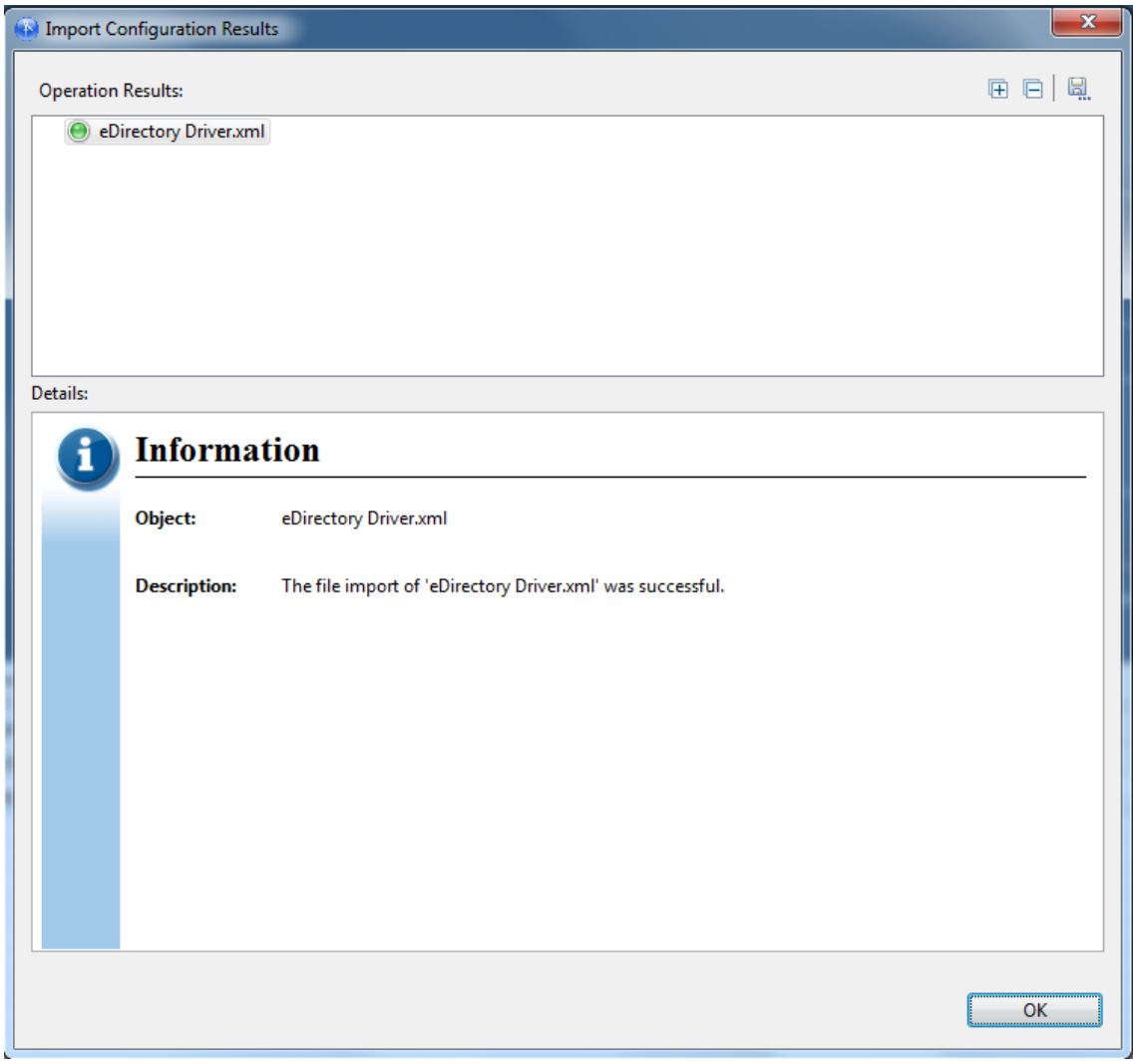
4 In the Perform Prompt Validation window, you are asked if you want to fill in required driver information. If you answer **Yes**, you see the Import Information Requested dialog box as described in Step 5 and you must provide values to all of the required fields. If you answer **No**, you still see the Import Information Requested dialog box, but it isn't necessary to fill in the required information.

5 Complete the import by filling out the Import Information forms for the driver configuration file as necessary.

Policies are saved with a skeleton driver configuration structure, which designates where the policy resides. In this case, the driver already existed and the imported policy was initially written for that driver.

6 Click **OK**.

When the policy or rule is imported, you see the Import Configuration Results window, which indicates if there were any problems with the import procedure.



Errors during the import procedure are displayed with a red icon, and you see an Error description that is related to the operation results. If you have multiple errors, selecting an error displays the error's description in the **Details > Description** field. See "Error Messages and Solutions" on page 543 for further information.

7 Click **OK** to finish the import process and close the Import Configuration Results window.

Importing Channels, Policies, and Schema Items from the Identity Vault

A channel is a combination of rules and policies, and Designer allows you to import a channel instead of the entire driver. The Subscriber and Publisher channels describe the direction in which the information flows. The Subscriber channel takes the event from the Identity Vault and sends that event to the receiving system (application, database, CSV file, etc.) The Publisher channel takes the

event from the application, database, CSV file, etc., and sends that event to the Identity Vault. The Subscriber and Publisher channels act independently; actions in one are not affected by what happens in the other.

- "Importing a Channel" on page 308
- "Importing a Policy" on page 310
- "Importing a Schema" on page 312

Importing a Channel

To import an OpenText Identity Manager channel (a Subscriber channel or a Publisher channel) object and all contained policies into a driver:

1. Select either a Driver object or an Application object in the Modeler view.

   The Driver object is represented by the line between the Identity Vault and the Application object and has a circle icon to represent it . The Application object connects to the Identity Vault through the Driver object.

2. Right-click a Driver object, then click **Live > Import**.

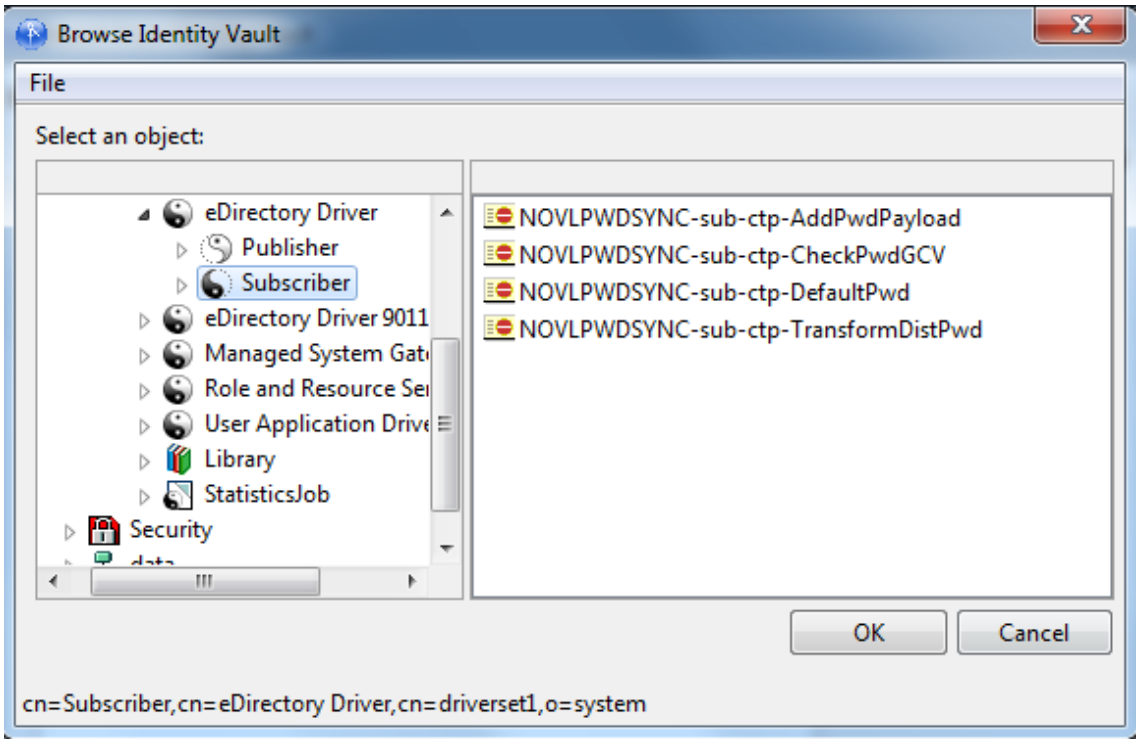
   or

   Right-click an Application object and click **Driver > Import**.

   If Designer can't authenticate to the OpenText eDirectory tree specified in the Identity Vault, you see the Identity Vault Credentials window if you have not previously specified the authentication credentials or if you do not save the password.

3. Fill in the appropriate information and click **OK**.

4. In the Import from Identity Vault window, browse to and select either a Publisher or a Subscriber Channel object from the OpenText eDirectory tree under the corresponding driver.

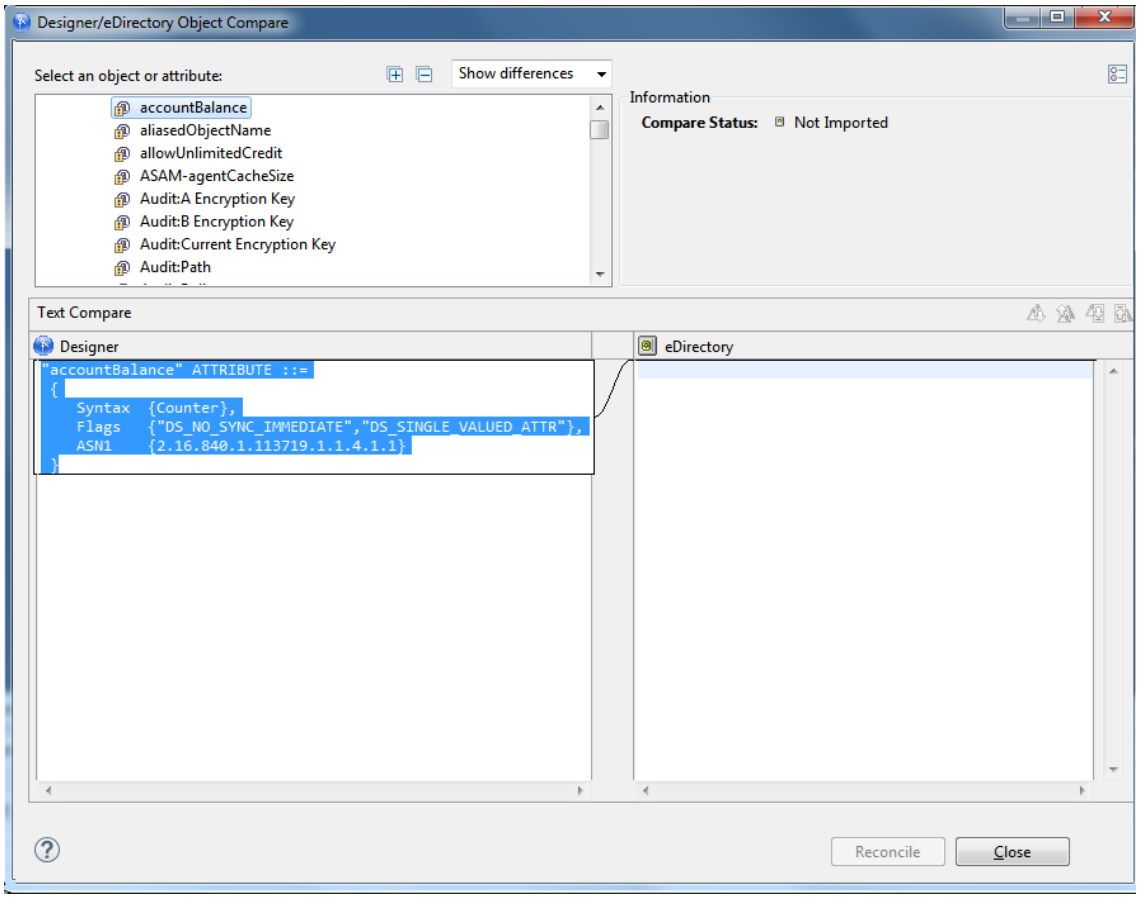


5 You can import more than one channel at a time; if you want to import both channels, select one channel, click **OK**, then browse to the next channel, select it, and click **OK**.

6 Click **Continue**.

As the channel imports, you see the Import Summary window showing you the differences between OpenText eDirectory (the source of the import) and Designer (the destination).

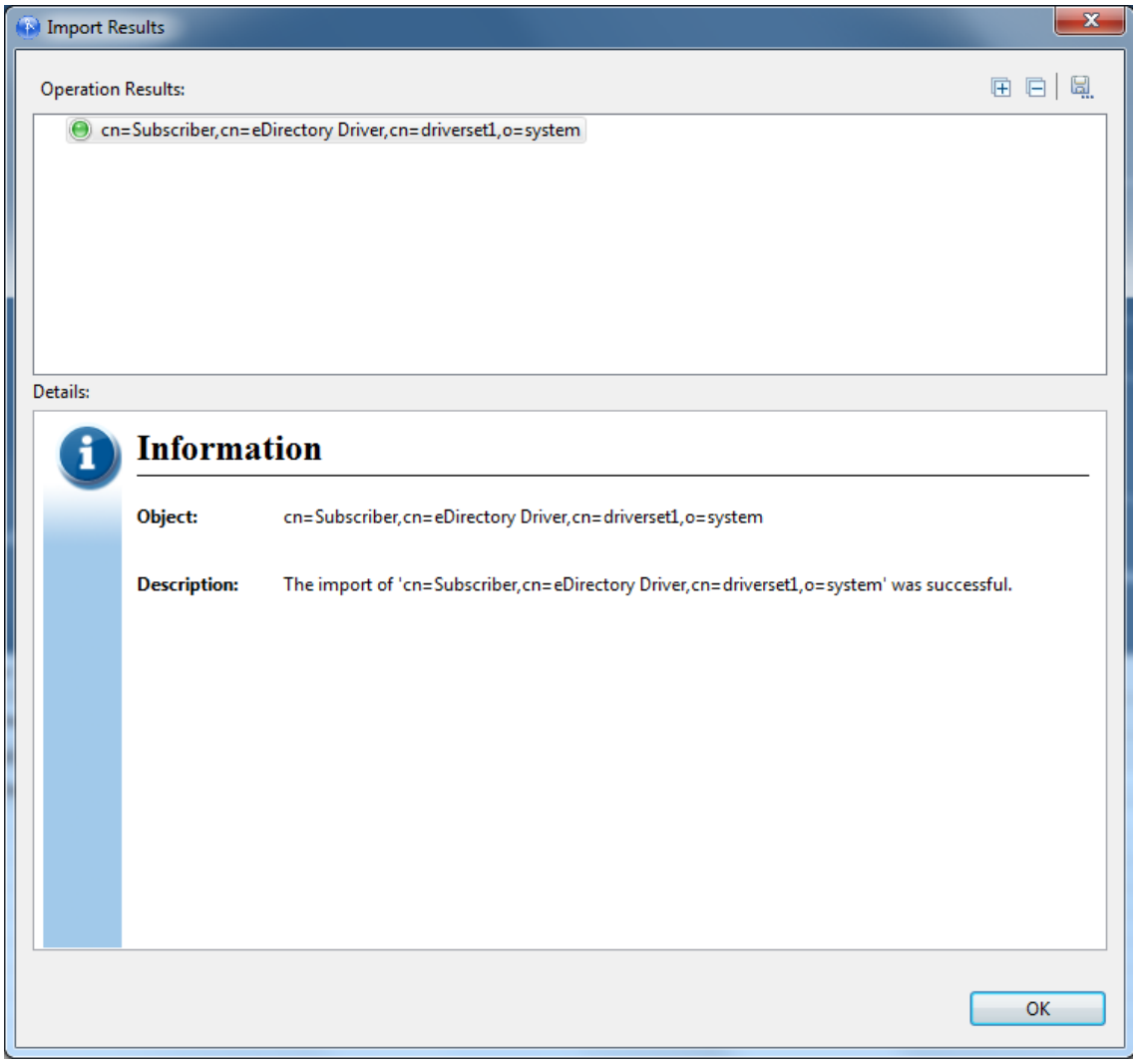
Figure 11-1



You can click the different objects in the channel view differences between the two drivers. All channel information is overwritten by the import procedure; however, the rest of the driver is unaffected.

7 Click **Import**.

For more information about Compare, see "Using the Compare Feature When Importing" on page 315.



If there are any problems with the import procedure, they are displayed with a red icon in the Import Results window and you see an error description that is related to the operation results. If you have multiple errors, selecting an error displays the error's description in the **Details > Description** field. See "Identity Vault Configuration Errors" on page 544 for further information.

8 Click **OK** to finish the import process.

Importing a Policy

A policy is a collection of rules and arguments that allows you to transform the data that an application sends to and receives from OpenText eDirectory. You use policies to manipulate the data you receive from OpenText eDirectory or from the managed system so they can synchronize the information in their databases. Each driver connects to a different system, and policies tell the driver how to synchronize the data on that managed system to the Identity Vault.

You might use the Import feature for policies more than anything else. For example, you can set up a policy to allow users with the title "Manager" to be placed in a specific container, no matter which application the information is coming from, and you can place this information in multiple managed systems. However, because each application is different, you need to modify the arguments and

rules within policies to reflect those differences. For more information about policies, see *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide* and *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

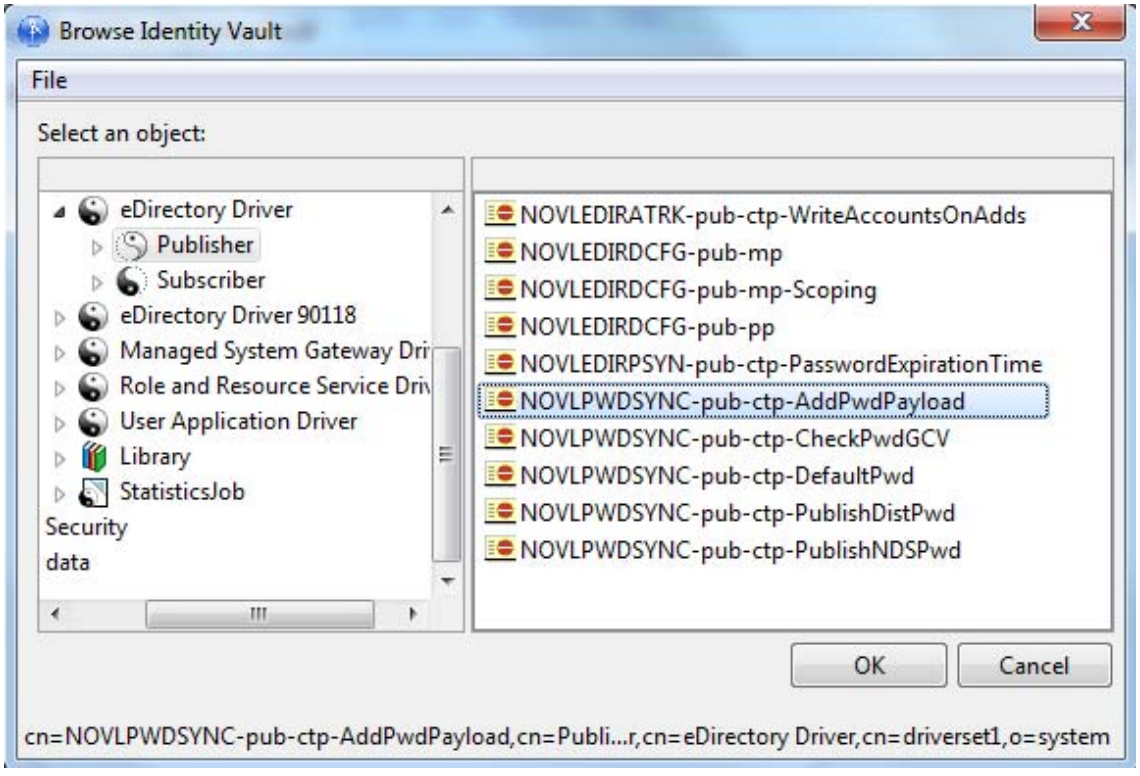
To import an OpenText eDirectory Policy object (for example, a rule or a style sheet) into a driver or channel (Subscriber or Publisher):

1. Select a driver in the Modeler view.

   or

   Click the **Outline** tab and select a Driver or Channel object from the Outline view.
2. Verify that the authentication credentials in the Properties view for the selected Identity Vault are correct.
3. Right-click the Driver or Channel object, then click **Live > Import**.

   If the application can't authenticate to the tree, you see the Identity Vault Credentials window asking for the hostname, username, and password if you have not previously specified the authentication credentials or if you do not save the password.
4. Fill in the appropriate information and click **OK**.
5. In the Import from Identity Vault window, click **Browse**, then select a policy object from the channel you specified when you started the import process.



   Policies are found under either the Publisher or Subscriber channel of a selected driver or under the driver itself. Be sure to match the proper policy to the proper channel or driver object.
6. Click **OK**, then click **Continue** to import the policy.

You see the Import Summary window showing you the differences between OpenText eDirectory (the source of the import) and Designer (the destination). You can click the different objects in the policy to see what is different between the two policies. All selected policy information is overwritten by the import procedure; however, the rest of the driver is unaffected.

7 Click **Import**.

If the importing policy contains the same values as the policy in Designer, you are not allowed to import the policy. (For more information about the Compare feature, see "Using the Compare Feature When Importing" on page 315.)

Clicking **Import** brings up the Import Results window. If there are any problems with the import procedure, they are displayed with a red icon, and you see an Error description that is related to the operation results. If you have multiple errors, selecting the different errors displays the error's description in the **Details > Description** field. See "Error Messages and Solutions" on page 543 for further information.

8 Click **OK** to finish the import process.

For policy design, see the Policy Builder and Policy Management Help topics within the Designer utility. Also see *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide* and *OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies*.

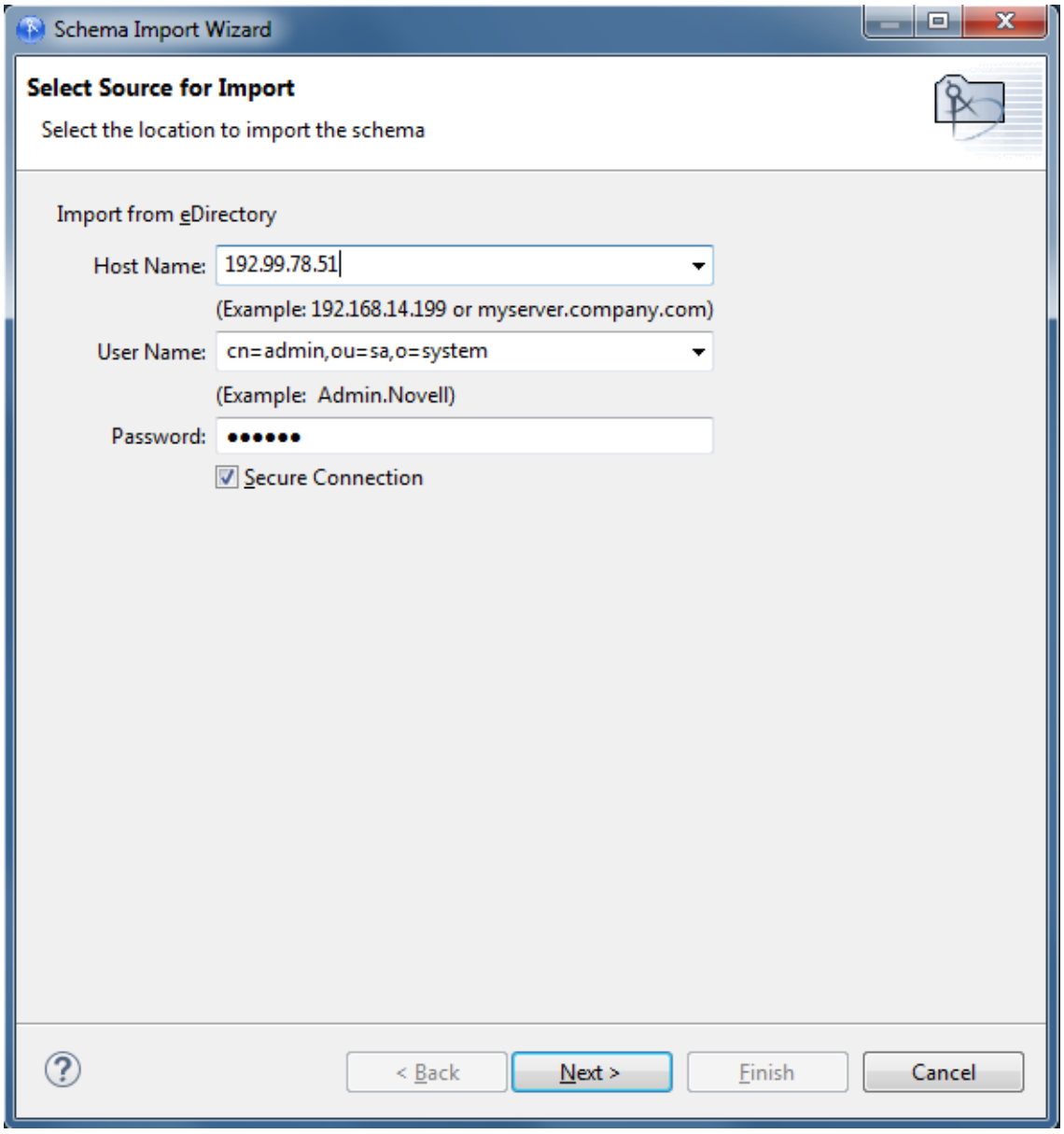
Importing a Schema

You can import a schema from the Identity Vault or from a .sch file into your project. When you import a schema, you can select the whole Identity Vault schema (not recommended) or just the schema differences between the Identity Vault and your project.

1 Bring up the project in Designer's Modeler view. Right-click the Identity Vault and select **Live > Schema > Import**.

2 On the Select Source for Import page, select **Import from eDirectory** if you can connect to an actual Identity Vault.

The specified user must have administrative rights to the schema.



3 In the Import from eDirectory section, specify the hostname, username, and password connection information.

The **Host Name** and **User Name** entries have drop-down menus storing the last information you typed into these fields, which you can use for filling in these entries.

4 Fill in the **User Name** and the **Password** fields.

You can use history drop-down lists to choose a previously entered value from a list.

5 When you enable a **Secure Connection** between Designer and the Identity Vault, the LDAP server listens on the default secure port (636). If you configured the secure port as `700`, specify this port number in the **Host name** field. For example, `192.99.78.51:700`.

When connecting through a secure port, Designer prompts you to import the Identity Vault's Certificate Authority certificate into Designer. You must accept this certificate to establish a secure connection with the Identity Vault. For more information, see "Secure Connection" in Table 3-2 on page 71.

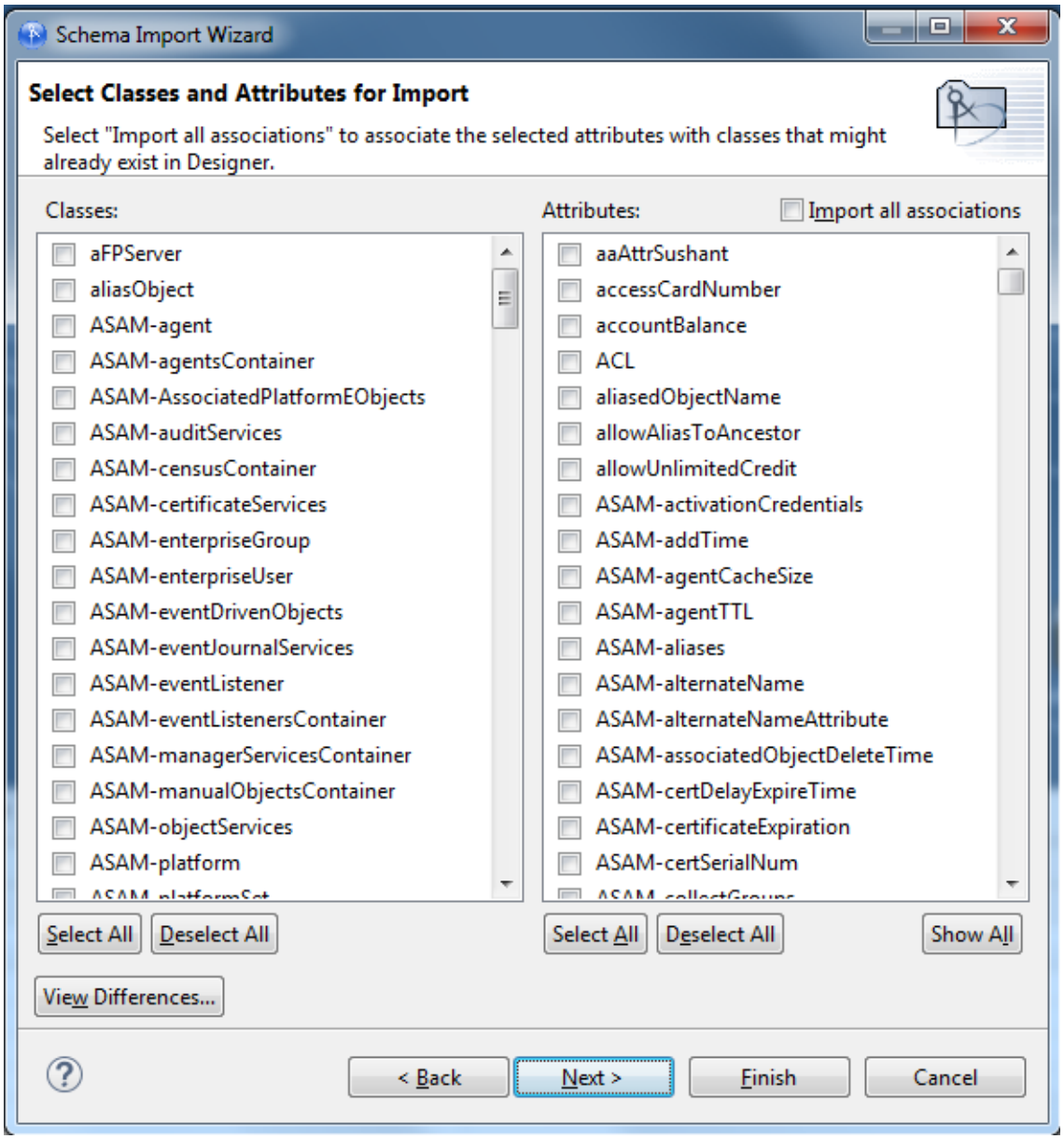
6 Click **Next**.

7 Decide which classes and attributes to import.

On the Select Classes and Attributes for Import page, you can select all of Identity Vault's schema, including classes and attributes. However, this can create very large documents when you document the project (600 pages or more).

8 If you want to import all the classes and attributes, click **Select All**, click **Finish**, then skip to Step 10.

Select only the classes and attributes that you want to import. If you only want to import the schema differences between the live Identity Vault and the Identity Vault in your project, click **View Differences**, then continue with Step 9.



On the Schema Differences page, you see the schema differences between the live Identity Vault and the Identity Vault in your project.

9 Click **Select All** if you only want to import schema differences. Otherwise, click **Cancel**.

10 Do one of the following options:

- Selecting **Select All > OK** brings you back to the Select Classes and Attributes for Import page with the schema differences now selected under the Classes and Attributes headings. If you select any classes from the Schema Differences page, the **Import all associations** box is selected. Leave it selected, because it enables you to associate the selected attributes with the classes that might already exist in Designer. Click **Finish**.
- If you selected **Cancel** on the Schema Differences page, make your schema selections on the Select Classes and Attributes for Import page, select the **Import all associations** box (recommended), and click **Finish**.
- Click **Next** if you want to see the Import Summary page to see the classes and attributes that you are importing. Then click **Finish**.

11 On the Import Messages page of the Schema Import Wizard, click **OK**.

or

If you want to save the differences to a log file, click **Save to Log**. This brings up the Save As dialog box, where you can choose a filename and directory to store the file in.

12 Click **Save**, then click **OK**.

Using the Compare Feature When Importing

The Compare feature allows you to see differences between the driver sets, drivers, channels, and policies that are stored in projects and those that are running in deployed systems, and reconcile any differences to either Designer or Identity Vault.

Designer provides conflict resolution on an object-by-object basis and allows you to view the differences between existing and new values when importing and deploying driver sets, drivers, channels, and policies. For example, before importing a driver object in Designer to a driver object that already exists in the Identity Vault, you can run Compare. Compare shows whether the driver objects are equal (no action is necessary) or unequal. If they are unequal, you can choose not to reconcile the driver objects, choose to update the driver object in Designer, or choose to update the driver object in the Identity Vault.

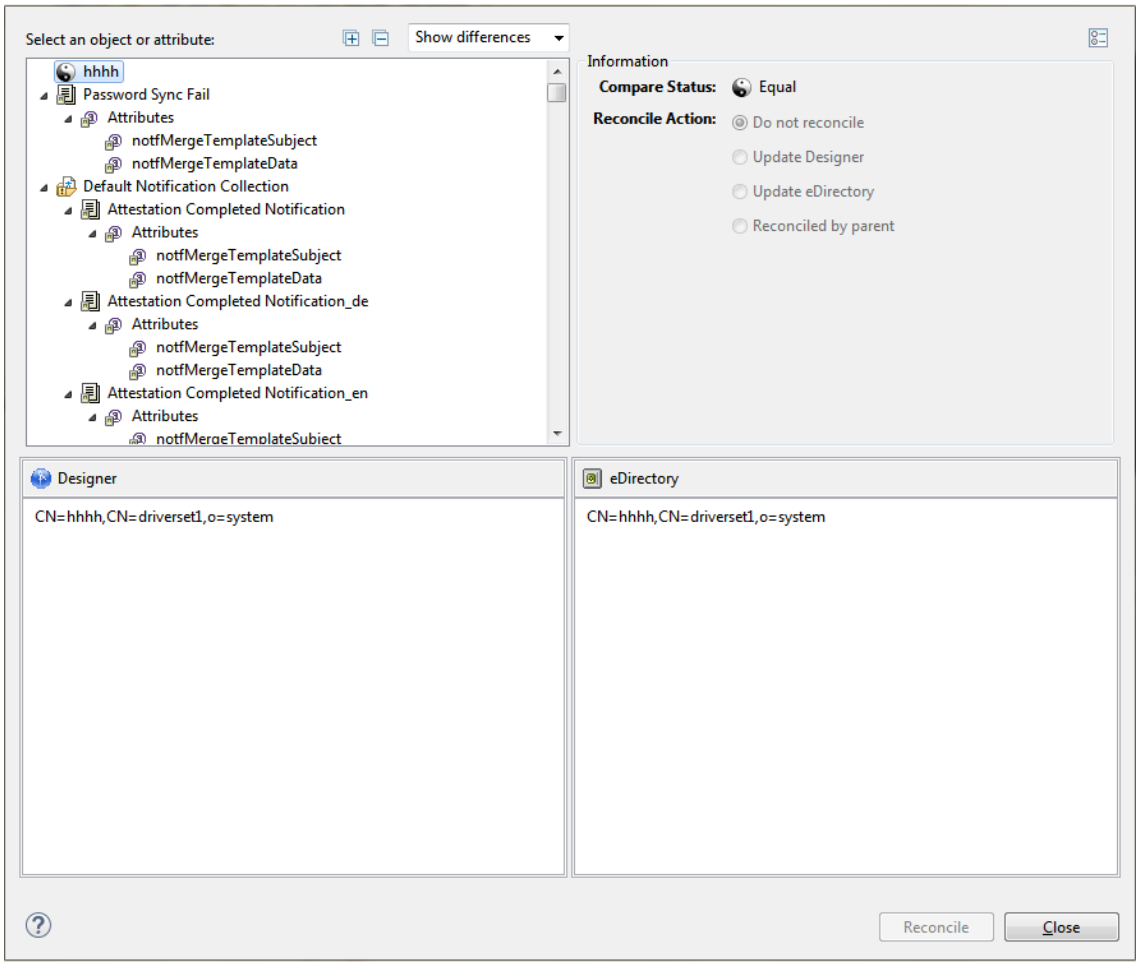
You can run the Compare feature at any time. If you choose to reconcile the differences between drivers objects in Designer and OpenText eDirectory while in Compare, you won't need to run Import or Deploy.

- "Using Compare When Importing a Driver Object" on page 316
- "Using Compare on a Channel Object" on page 318
- "Using Compare on a Policy" on page 318
- "Matching Attributes with Designer Properties" on page 318

Using Compare When Importing a Driver Object

Use this procedure if you want to import a Driver object from the Identity Vault and the same driver already exists in Designer.

1 Right-click the driver object in either the Modeler view or in the Outline view, then click **Live > Compare** to bring up the Designer/OpenText eDirectory Object Compare window.



Under the **Select an object or attribute**, you see the selected object with the differences between Designer's and OpenText eDirectory's driver object. You can select the attributes and child objects to see the actual differences displayed in the **Text Compare** area.

The plus icon at the right side of the **Select an object or attribute** allows you to expand all elements in the parent object, and the minus icon collapses all of the elements. The "?" icon in the bottom left portion of the window displays the Summary/Compare dialog box help.

Server-specific attributes are attributes that have a value for each server that is associated with a driver set. These attributes are displayed in the **Attributes** list with the server name in parentheses to the right of the attribute name.

2 By default, the Compare window only displays values that are different between Identity Vault and Designer. To view all of the object values, select **Show All** from the drop-down menu.

Values that are equal are shown as **Equal** on the **Compare Status** line under **Information**.

The overlay image displayed in the **Compare Status** entry identifies objects or attributes that need reconciliation. The following table describes what you see in the **Compare Status** line and the overlays that you can see:

Compare Status	Description
Equal	The selected attribute's value or all attributes of the selected object are the same in OpenText eDirectory and Designer.
Unequal 	The value of the selected attribute, or one or more attributes of the selected object, are different in OpenText eDirectory and Designer.
Not Deployed	The selected object or the object containing the selected attribute is not deployed to OpenText eDirectory.
Not Imported	The selected object or object containing the selected attribute does not exist in Designer.
Unknown	The selected object or object containing the selected attribute cannot be compared, such as a password.
Deleted	Designer tracks objects that are deployed, then deleted from the Designer project.

You can also see an **Attribute Note** if you select an attribute.

3 In the **Information** portion of the Compare window, select how you want to reconcile the differences between the Source and Destination. If **Compare Status** shows **Unequal**, you have three choices:

- **Do not reconcile:** To do nothing, keep the default value of **Do Not Reconcile**.
- **Update Designer:** To update the driver in Designer so that it contains the same information as the driver in the Identity Vault, select **Update Designer**.
- **Update eDirectory:** To update the driver in OpenText eDirectory to reflect the changes you have just made to the driver in Designer, select **Update eDirectory**.

If you select the parent object to perform the update, then all of the child objects under the parent reflect that choice and you see the **Reconciled By Parent** icon selected. If you do not choose a parent object, you can reconcile each child object individually.

4 View the differences displayed in the Text Compare area.

The Text Compare values displayed in the bottom portion of the Designer/OpenText eDirectory Object Compare window vary, depending on the object being compared. For instance, Compare shows you changes down to the policy level. The Text Compare dialog box uses the Eclipse Compare editor to compare attributes that contain XML data, such as policy data, driver filters, or configuration data. The differences in the code are highlighted in blue.

5 After you view the differences, click **Reconcile** to perform the reconciliation actions for each object in the tree, or click **Close** to close the Designer/OpenText eDirectory Object Compare screen.

After reconciliation, the object matches both locations and has been imported or deployed through the action.

Using Compare on a Channel Object

Use this procedure if you want to import a channel object from the Identity Vault and the same channel already exists in Designer. You can view the differences and decide whether to reconcile them.

1 Right-click the channel object in the Outline view. Click **Live > Compare** to bring up the Designer/eDirectory Object Compare window.

All Compare windows behave as described in "Using Compare When Importing a Driver Object" on page 316.

After reconciliation, the Channel object matches both locations and has been imported or deployed through the action.

Using Compare on a Policy

Use this procedure if you want to import a policy object from the Identity Vault and the same channel already exists in Designer. You can view the differences and decide whether to reconcile them.

1 Right-click the policy object in the Outline view. Select **Live > Compare** to bring up the Designer/eDirectory Object Compare window.

All Compare windows behave as described in "Using Compare When Importing a Driver Object" on page 316.

After reconciliation, the policy object matches both locations and has been imported or deployed through the action.

Matching Attributes with Designer Properties

The attributes of the object are displayed in the single list.in the compare window. Selecting an attribute displays its value below the attribute list with the Designer value on the left and the OpenText eDirectory value on the right. The name displayed in the list is the OpenText eDirectory attribute name.

The following tables map the OpenText eDirectory attribute to the Designer property page or control where you can change or set the attribute (you can't make changes inside the Compare window).

- Table 11-1 on page 319 shows Driver Set OpenText eDirectory attributes
- Table 11-2 on page 319 shows Driver OpenText eDirectory attributes
- Table 11-3 on page 320 shows Channel OpenText eDirectory attributes
- Table 11-4 on page 321 shows the Job OpenText eDirectory attributes
- Table 11-5 on page 321 shows the Resource OpenText eDirectory attributes

- Table 11-6 on page 321 shows the ID Policy OpenText eDirectory attributes
- Table 11-7 on page 322 shows the Library OpenText eDirectory attribute
- Table 11-8 on page 322 shows the Notification Template OpenText eDirectory attributes
- Table 11-9 on page 322 shows the Notification Template Collection OpenText eDirectory attributes

Table 11-1 *Driver Set OpenText eDirectory Attributes*

Driver Set OpenText eDirectory Attribute	Designer Property
DirXML-DriverTraceLevel	Driver Set Properties > Trace > Driver Trace Level
DirXML-XSLTraceLevel	Driver Set Properties > Trace > XSL Trace Level
DirXML-JavaEnvironmentParameters	Driver Set Properties > Java
DirXML-JavaDebugPort	Driver Set Properties > Trace > Java Debug Port
DirXML-JavaTraceFile	Driver Set Properties > Trace > Java Trace File
DirXML-Trace File Encoding	Driver Set Properties > Trace - Trace File Encoding
DirXML-TraceSizeLimit	Driver Set Properties > Trace > Trace File Size Limit
DirXML-LogLimit	Driver Set Properties > Driver Set Log Level > Log Limit
DirXML-LogEvents	Driver Set Properties > Driver Set Log Level > Log Specific Events
DirXML-NamedPasswords	Driver Set Properties > Named Passwords
DirXML-ConfigValues	Driver Set Properties > Global Configuration Values

Table 11-2 *Driver OpenText eDirectory Attributes*

Driver OpenText eDirectory Attribute	Designer Property or View
DirXML-InputTransform	Policy Set View > Input Transformation
DirXML-OutputTransform	Policy Set View > Output Transformation
DirXML-MappingRule	Policy Set View > Schema Mapping
DirXML-Driver Filter	Policy Set View > Driver Filter
DirXML-ConfigValues	Driver Properties > Global Configuration Values
DirXML-DriverTraceLevel	Driver Properties > Driver Log Level > Driver Log Level
DirXML-EngineControlValues	Driver Properties > Engine Control Values
DirXML-LogEvents	Driver Properties > Driver Log Level > Log Specific Events
DirXML-LogLimit	Driver Properties > Driver Log Level > Log Limit
DirXML-ConfigManifest	Driver Properties > Driver Manifest

Driver OpenText eDirectory Attribute	Designer Property or View
DirXML-JavaModule	Driver Properties > Driver Configuration > Driver Module: Java
DirXML-NativeModule	Driver Properties > Driver Configuration > Driver Module: Native
DirXML-DriverImage	Driver Properties > iManager Icon
DirXML-ReciprocalAttrMap	Driver Properties > Reciprocal Attributes
DirXML-TraceLevel	Driver Properties > Trace > Trace Level
DirXML-TraceFile	Driver Properties > Trace > Trace File
DirXML-TraceFileEncoding	Driver Properties > Trace > Trace File Encoding
DirXML-TraceSizeLimit	Driver Properties > Trace > Trace File Size Limit
DirXML-TraceName	Driver Properties > Trace > Trace Name
DirXML-DriverCacheLimit	Driver Properties > Driver Configuration > Authentication > Driver Cache Limit
DirXML-ShimAuthID	Driver Properties > Driver Configuration > Authentication > User ID
DirXML-ShimAuthServer	Driver Properties > Driver Configuration > Authentication > Connection Information
DirXML-ShimAuthPassword	Driver Properties > Driver Configuration > Authentication > Set Password
DirXML-ShimConfigInfo	Driver Properties > Driver Configuration > Driver Configuration > Driver Parameters
DirXML-DriverStartOption	Driver Properties > Driver Configuration > Startup Option
DirXML-ECMAScript	Driver Properties > Driver Configuration > ECMAScript
DirXML-NamedPasswords	Driver Properties > Named Passwords

Table 11-3 *Channel OpenText eDirectory Attributes*

Channel OpenText eDirectory Attribute	Designer View
DirXML-EventTransformationRule	Policy Set View > Event Transformation
DirXML-MatchingRule	Policy Set View > Matching
DirXML-CreateRule	Policy Set View > Creation
DirXML-PlacementRule	Policy Set View > Placement
DirXML-CommandTransformation	Policy Set View > Command Transformation

Table 11-4 *Job OpenText eDirectory Attributes*

Job OpenText eDirectory Attribute	Designer View
XmlData	Job Editor, XML cannot be edited directly only through Job Editor UI
DirXML-ServerList	Job Editor
DirXML-Scope	Job Editor
DirXML-EMailTemplates	Job Editor
DirXML-EMailServer	Job Editor
DirXML-NamedPasswords	Job Editor
DirXML-TraceName	Job Properties > Trace
DirXML-TraceFile	Job Properties > Trace
DirXMl-TraceSizeLimit	Job Properties > Trace
DirXML-TraceFileEncoding	Job Properties > Trace > Trace File Encoding
DirXML-TraceLevel	Job Properties > Trace

Table 11-5 *Resource OpenText eDirectory Attributes*

Resource OpenText eDirectory Attribute	Designer View
DirXML-ContentType	Read only, cannot be edited set at creation time of the object
DirXML-DirXMLData	Resource Editor
DirXML-NamedPasswords	Resource Editor

Table 11-6 *ID Policy OpenText eDirectory Attributes*

ID Policy OpenText eDirectory Attribute	Designer View
DirXML-idPolMin	ID Policy Properties > Constraints Minimum
DirXML-idPolMax	ID Policy Properties > Constraints Maximum
DirXML-idPolPrefix	ID Policy Properties > Constraints Prefix
DirXML-idPolArea	ID Policy Properties > Constraints Exclude/Include Text Field
DirXML-idPolFill	ID Policy Properties > Constratints Fill Yes/No
DirXML-idPolAreaEI	ID Policy Properties > Constraints Exclude/Include Radio Button
DirXML-idPolAccessControl	ID Policy Properties > Access Control enabled
DirXML-idPolACL	ID Policy Properties > Access Control ACL

Table 11-7 *Library OpenText eDirectory Attribute*

Library OpenText eDirectory Attribute	Designer View
Description	Library Properties > Description

Table 11-8 *Notification Template OpenText eDirectory Attributes*

Notification Template OpenText eDirectory Attributes	Designer View
notfMergeTemplateSubject	Template Editor
notfMergeTemplateData	Template Editor

Table 11-9 *Notification Template Collection Attributes*

Notification Template Collection Attributes	Designer View
notfSMTPEmailHost	Notification Template Collection Properties > Host Name
notfSMTPEmailFrom	Notification Template Collection Properties > From
notfSMTPEmailUserName	Notification Template Collection Properties > User Name

Error Messages and Solutions

To view error messages along with their possible solutions associated with importing and deploying files, see “Deploying OpenText Identity Manager Objects” on page 536.

12 Documenting Projects

When you create a project, it's vital to keep track of how the driver works and how it's implemented into a network. The Document Generator helps you quickly generate customized documentation for your Designer projects. These documents can save you weeks or months of gathering and writing driver specifications and their implementations. To generate a document, choose a document style (it can be the default style that comes with Designer or one that you customize) and a Designer project or portion of a project. The Document feature combines the information and structure of the selected style with the project information in order to generate customized project documentation.

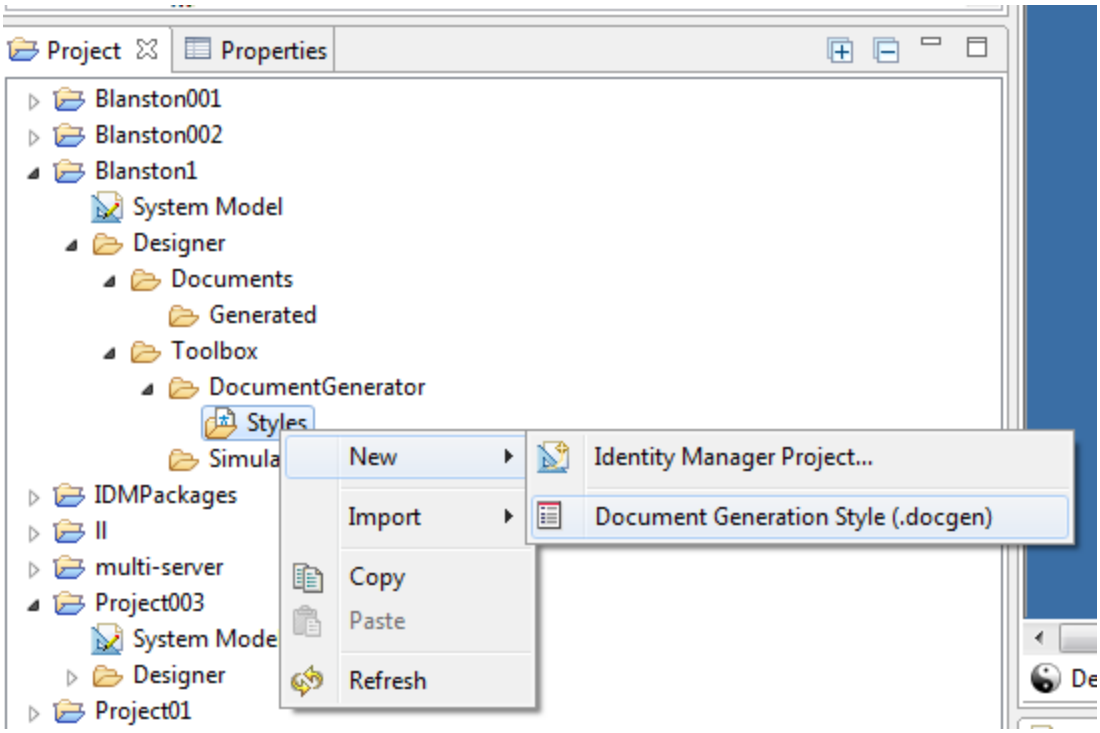
Designer comes with a default document style so you won't need to create a document from scratch. This default style contains everything that you have placed in a project through Designer. You must first use this default style to create your own document style for the project you are working on, then you can either use it as it is or customize it to meet your particular needs, including or excluding information as needed. After you have edited the style to your liking, you can also use it to document your other projects. There is an advanced editing feature that allows you to create your own sections for adding information that you did not create in Designer.

- "Creating a Document Style" on page 323
- "Editing a Document Style for Your Needs" on page 326
- "Generating a Document" on page 327
- "Using Your Style Template for Other Projects" on page 328
- "Customizing Styles to Include or Exclude Information" on page 330
- "Advanced Editing of a Document Style" on page 333

Creating a Document Style

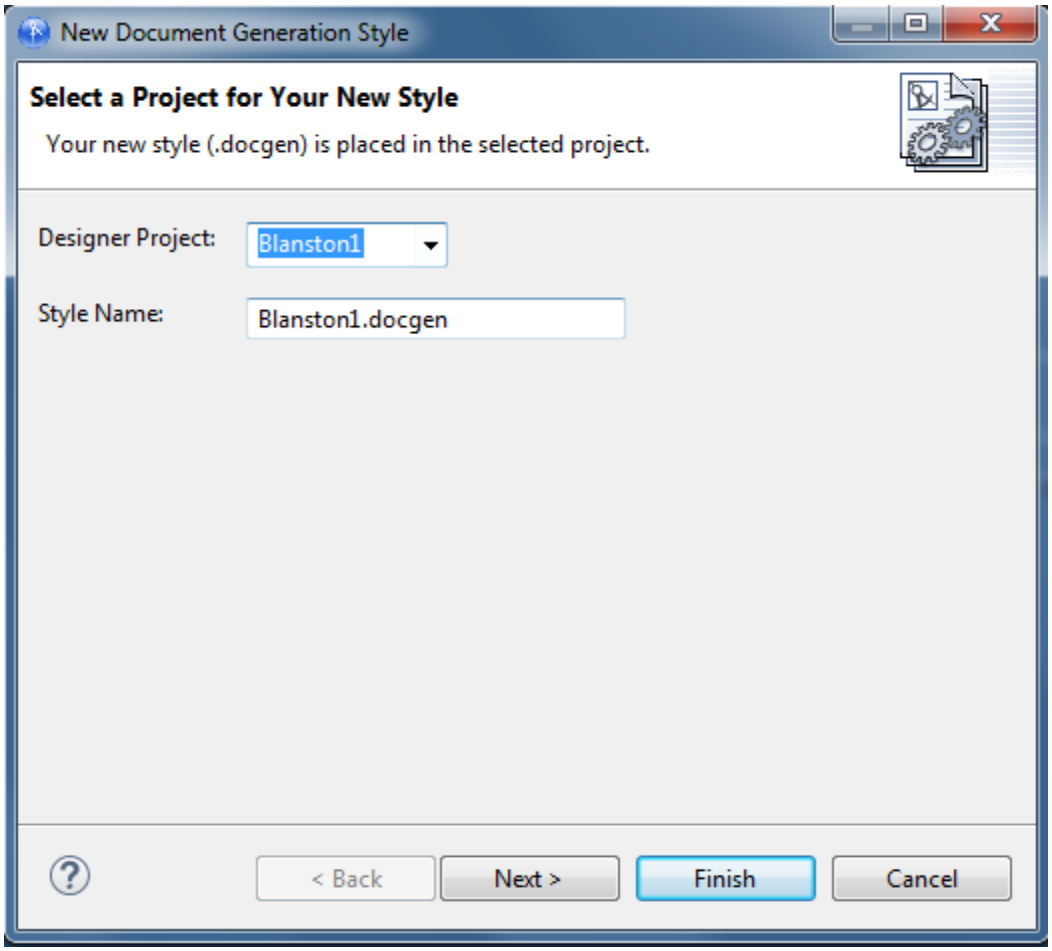
A document style allows you to define how you want your project information to look. In a matter of minutes, you can generate a document that contains all elements that you have placed in a project and define a document style to designate how the information looks, as well as what information you want in a document.

1. Select a project in the Project view, then right-click the **Toolbox > DocumentGenerator > Styles** icon.
2. Select **New > Document Generation Style (.docgen).**



The Document Generation Style (`.docgen`) is the default style that is provided with Designer. You use this as the template for your own `.docgen` style.

3 Specify a name for the Designer project, or use the default of the project's name. Then specify a name for the style, with a `.docgen` extension, or use the default name of the project you are presently in, then click **Finish**.



A `.docgen` file is created that you can use as the basis for designing your own style template. The `.docgen` template is placed in the Style Editor view for your modification (see "Editing a Document Style for Your Needs" on page 326). You can use the defined elements in your new style template again and again.

4 Use this basic document style template to generate documentation for a project, or customize it for your needs.

- "Generating a Document" on page 327
- "Editing a Document Style for Your Needs" on page 326
- "Using Your Style Template for Other Projects" on page 328
- "Customizing Styles to Include or Exclude Information" on page 330
- "Advanced Editing of a Document Style" on page 333

Editing a Document Style for Your Needs

After you have created a .docgen style template, you can edit the file to meet your documentation needs.

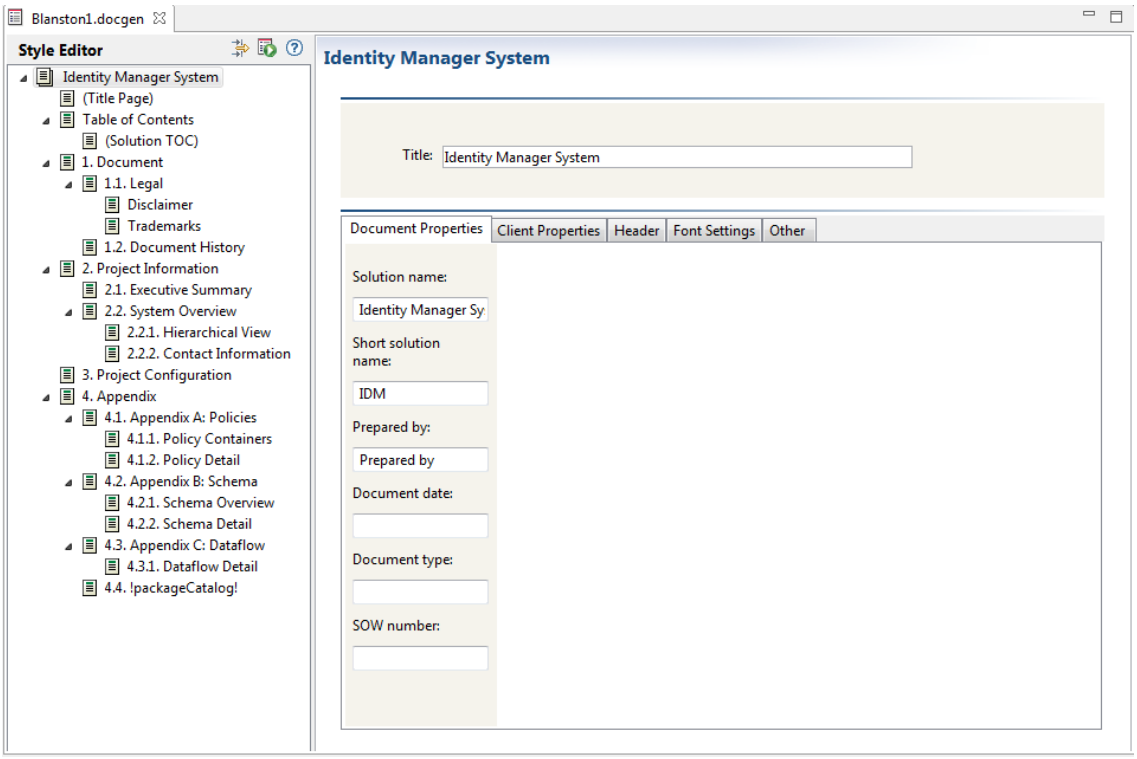
You can add or modify information in the style template, which in turn affects the documents that you generate. The information that you can customize through the Style Editor appears in the Style Editor view. For more detailed editing, see “Advanced Editing of a Document Style” on page 333.

- “Editing a Style Template” on page 326
- “Editing Sections of a Style” on page 327

Editing a Style Template

1 Select a project in the Project view, then expand the **Designer > Toolbox > DocumentGenerator > Styles** icon.

2 Double-click the .docgen file. The file appears in the Style Editor view.



3 Use the Style editor to edit sections of a style or to modify the style according to your needs.

The Style editor is divided into two parts, beginning with the OpenText Identity Manager and working through the Appendixes. When you click an item under the **Style Editor** section, the right portion of the view changes to display the information associated with the heading. For example, clicking **Disclaimer** under the **Document > Legal** heading allows you to edit the disclaimer content.

4 Save your changes. Your changes are saved when you close the Style Editor, or when you click the **Save** icon .

Editing Sections of a Style

1. Select an item (for example, Executive Summary) in the Style Editor view.
2. Enter data or make changes in the left pane.
3. Select other items as appropriate and make changes. The information in the left pane varies, depending on items that you select.

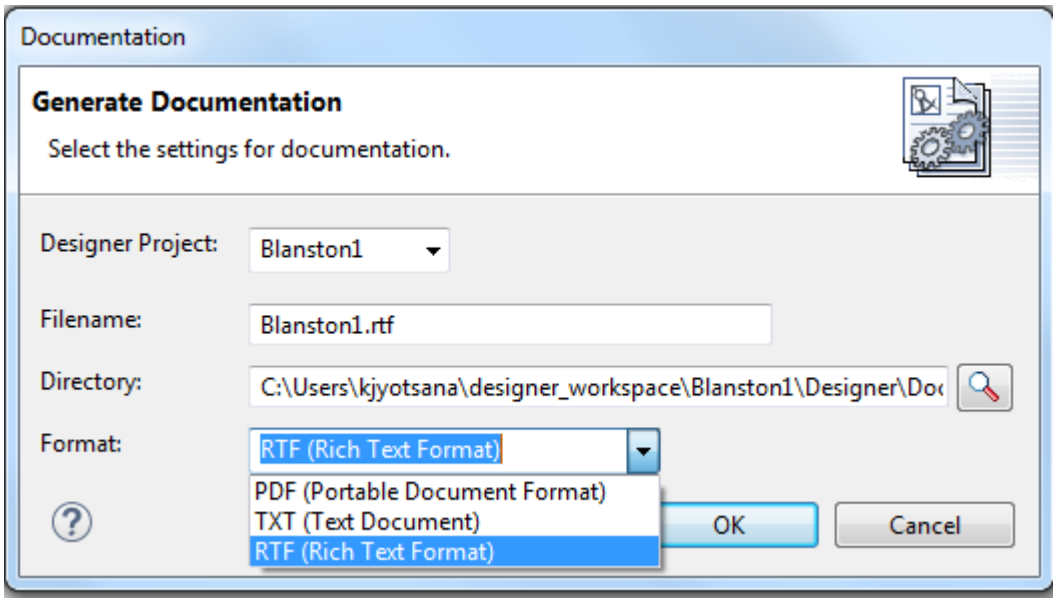
   The main areas that you need to pay attention to are the information found under OpenText Identity Manager System (Title Page and Table of Contents), Document, Legal, Disclaimer, Trademarks, and Executive Summary.
4. Save your changes. Your changes are saved when you close the Style Editor, or when you click **Save** .
5. Use this document style to generate documentation for a project, or continue to customize it for specific documentation needs.
   - “Generating a Document” on page 327
   - “Using Your Style Template for Other Projects” on page 328
   - “Customizing Styles to Include or Exclude Information” on page 330
   - “Advanced Editing of a Document Style” on page 333

Generating a Document

1. (Conditional) If you haven’t yet created a Designer project, create one.
   - **1a** Select **File > New > OpenText Identity Manager Project**.
   - **1b** Provide a project name, then click **Finish**.
2. (Conditional) If you haven’t yet created a document style that you want to use as a template for your documentation, create one. See “Creating a Document Style” on page 323 and “Editing a Document Style for Your Needs” on page 326.
3. In the Project view, select and expand a project, then right-click the .docgen icon under **Designer > Toolbox > DocumentGenerator > Styles** and select **Generate Documentation for This Style**.

   You can also expand the **Designer > Toolbox > DocumentGenerator > Styles** folder and click the .docgen file to open the file in the Style Editor, then click the **Document Generation** icon to the right of the Style Editor heading.
4. (Conditional) If the Project folder you selected does not contain a .docgen file, you are asked to select a Base Style. Select a .docgen style, then click **Next**.
5. (Conditional) Designer includes the ability to generate documents to RTF (Rich Text Format). If you want to enable this functionality, click **Window > Preferences** to bring up the Preferences window. Then, under **NetIQ > OpenText Identity Manager**, select **Document Generation**.

   By selecting **Enable RTF support (experimental)**, you can select the RTF format when creating documents.



6 On the Generate Documentation page, fill in the needed information.

- Select the name you want to call the file, or keep the default name. If you are generating the whole document, the default name often suffices. If you are generating a document for a section, such as an Executive Summary, name the file to reflect the section you are documenting.
- Select the directory where you want to store the document. If you use the default output location that appears in the Directory field, your generated document is visible under the `Documents\Generated` folder of the Project View.
- Select the format for the file. If you have enabled rich text formatting, you can select **PDF (Printable Document Format)**, **TXT (Text Document)**, or **RTF (Rich Text Format)**.

7 Click **Finish** to generate the document. The document appears in the current `Project > Documents > Generated` folder unless otherwise specified.

PDF files must be viewed through a PDF viewer, such as Adobe Acrobat. If Adobe Acrobat is installed on your workstation, Designer launches the document in Acrobat. RTF files can be viewed in any word processor that can handle Rich Text Formatted files, such as Wordpad in Windows.

The Filter editor provides an option to add notes to class and attributes, and these notes are added to the documentation. Password synchronization on drivers is also documented, showing how the administrator has set up password synchronization for the Publisher and Subscriber channels. You can also document contact information about the administrator for Identity Vault and application objects, as well as reciprocal mapping information.

Using Your Style Template for Other Projects

To generate documentation for any project, you can use the default style provided with Designer or you can use your own customized styles.

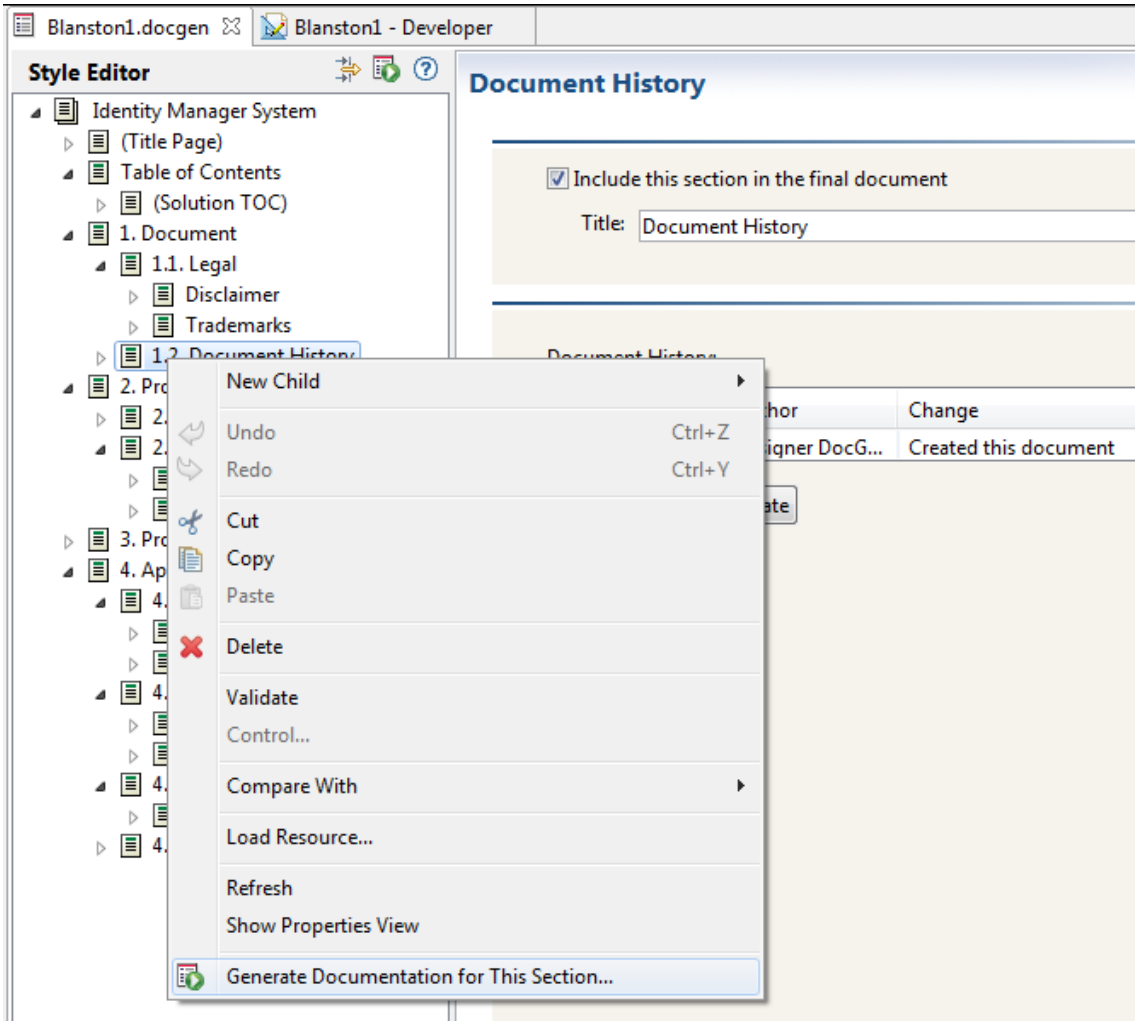
- "Documenting a Section of the Project" on page 329
- "Documenting Multiple Sections of the Project" on page 330

Documenting a Section of the Project

Instead of generating a document for the entire project, you can generate a document for a selected section of a project.

1 With the project's .docgen file open in the Style Editor pane, right-click a section of the style.

2 Select **Generate Documentation for This Section**.



3 In the Generate Documentation window, type a different project name in the **Filename** field (for example, DocHistoryofMerger), then click **OK**.

Specify which portion or portions of the project you want to include in the generated document.

You can document domains, Identity Vaults, driver sets, drivers, and applications using the Modeler view or the Outline view (use the Ctrl key to select multiple items). Document generation also ties in with schema notes, classes and attributes. You can find out more about this in Chapter 5, "Managing the Schema," on page 137.

For example:

1. To document a specific driver in a project, right-click the driver in the Modeler or Outline view and select **Document Selection**.
2. Select the .docgen style sheet for the document and click **Next**.
3. Give the document a filename, such as the driver's name, select the document's format, and click **Finish** to generate the driver document.

By default, documenting an application includes the connected driver (and its related objects). Likewise, documenting a specific driver includes its connected application. However, you can change this behavior in the Documentation Generation's Preferences page.

1. Click **Window > Preferences** to bring up the Preferences window.
2. Under **OpenText Identity Manager**, select **Document Generation**.

   Under the Modeler heading, the **Document applications and drivers related to other selected items** option is selected by default, which means that directly related items are included in the documentation. For example, by default, documenting a driver set includes the direct children (the applications) as well as some information of the direct parents (the Identity Vault and domain) to give context to the driver set. Deselecting this option excludes direct children of the selected item.
3. Select or deselect the options you want, then click **OK**.

Documenting Multiple Sections of the Project

If necessary, you can generate only selected sections so that peers can help you with information in the selected sections.

1. If you have not already done so, double-click the .docgen file to bring up the template in the Style Editor.
2. Select or deselect section headings. Each section and child section has a check box entitled **Include this section in the final document**. By default the box is selected, as shown below.

   To limit the sections you want to document, deselect the check boxes in the sections you don't want to generate.
3. Click the Generate Document icon  to the right of the Style Editor heading and generate your document.

   Give the document a unique name to reflect the type of information it includes.

Customizing Styles to Include or Exclude Information

We do not recommend that you document all Identity Vault schemas unless you need to.

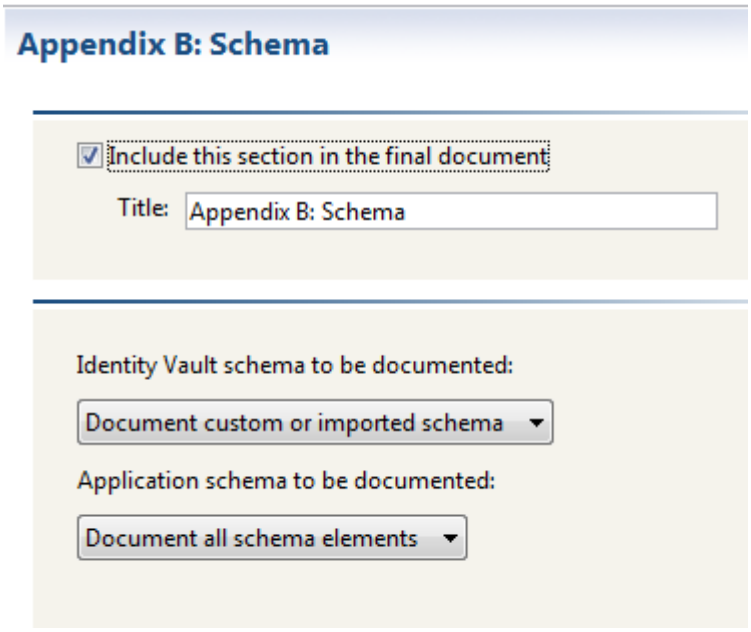
- "Identity Vault Schema and Application Schema" on page 331
- "Using Project Configuration to Limit Information" on page 331

Identity Vault Schema and Application Schema

The defaults for Identity Vault schemas have been changed to include custom schemas and any modified changes to the Identity Vault base schema. For application schemas, Designer includes all schemas by default. However, these can be turned off.

Select the **Appendix B: Schema** heading the in the Style Editor view. This brings up the Appendix B section template in the right side of the Style Editor view.

Figure 12-1 *The Appendix B: Schema Section Template*



The Appendix B: Schema section has three selections:

- **Include this section.** The **Include this section in the final documentation** check box allows you to include or not include Appendix B information in the documentation. By default, the box is selected to include this information. Deselect the check box if you do not want to include application or Identity Vault schemas in the document.
- **Document Custom and Imported Identity Vault Schema.** By default, the **Identity Vault Schema to be documented** selected documents any schema that you import from the Identity Vault or that you create. The choices are **Document custom or imported schema,** or **None**.
- **Document all Application Schema.** By default, the **Application schema to be documented** selection includes all of the application schema. The choices are **Document all schema elements**, or **None**.

Using Project Configuration to Limit Information

The Project Configuration heading allows you to include or deselect information about:

- Identity Vault
- Driver Sets

- Drivers
- General

The following table shows what type of information can be included or excluded in these areas.

Table 12-1 *Project Configuration*

Identity Vault	Information to Select or Deselect
Selected	Include host information
Selected	Include username information
Selected	Include deployment context information
Selected	Include driver set names
Selected	Include policy library on Identity Vaults
Deselected	Include e-mail notification templates
Deselected	Include XML source while documenting policies under the policy library
Deselected	Include XML source while documenting credential provisioning objects under the policy library
Deselected	Include XML source while documenting mapping table objects under the policy library
Driver Set	Information to select or deselect
Selected	Include server information associated with the driver set
Selected	Include driver set Global Configuration Value (GCV)
Selected	Include the policy library on driver sets
Selected	Include job objects on driver sets
Deselected	Include the XML source for policies under the policy library
Deselected	Include the XML source for credential provisioning objects under the policy library
Deselected	Include the XML source for mapping table objects under the policy library
Deselected	Include the XML source for job objects
Driver	Information to select or deselect
Selected	Include the driver filter policy
Selected	Include policy set
Selected	Include server-specific information for this driver
Selected	Include Remote Loader configuration
Selected	Include entitlements
Selected	Include credential provisioning
Selected	Include mapping table

Identity Vault	Information to Select or Deselect
Selected	Include ECMAScript resource object
Selected	Include job objects
Deselected	Include the XML source when documenting entitlement objects
Deselected	Include the XML source when documenting credential provisioning objects
Deselected	Include the XML source when documenting mapping table objects
Deselected	Include the XML source when documenting job objects
General	Information to select or deselect
Deselected	Include passwords
Selected	Page break after this section

IMPORTANT: Credential provisioning for the XML source might contain passwords that are displayed in clear text. If this option is selected, passwords are displayed in clear text and the documentation includes all passwords in the project.

Advanced Editing of a Document Style

In addition to selecting and deselecting the content of a document style, you can also change the layout and usability of your document style. You do this by editing the attributes that are associated with certain sections. You can also create additional sections for your documents as you see fit.

- "What's In the Advanced Editing Mode" on page 333
- "A Walk-through Tutorial" on page 340
- "Selecting a Language for Generated Documents" on page 346
- "Double-Byte Font Support" on page 347

What's In the Advanced Editing Mode

The Enable Advanced Editing icon lets you toggle between simple editing and advanced editing modes. By using the advanced editing mode, you can define information and a structure that is different from the default template already attached to a predefined style. In this example, the Title Page template is shown in its XSL format, which you must maintain.

Figure 12-2 *Viewing a Predefined Template*

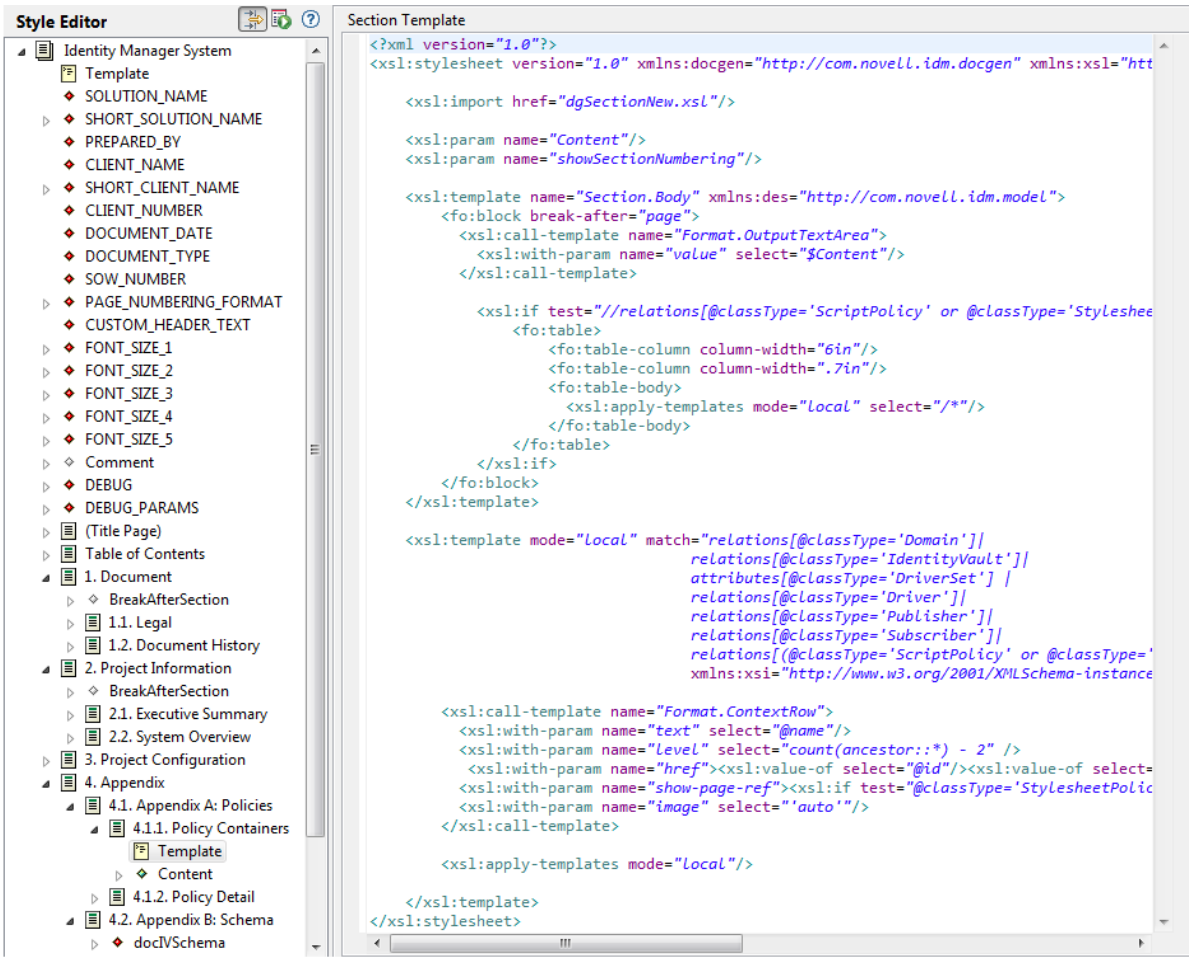


Table 12-2 *Style Editor Legend*

Icon	Name	Description
	Green Page	A green page means it's a titled section. The title appears when you generate the document.
	Grey Page	A grey page means it's not a titled selection. These pages are also in parentheses; for example, (Title Page).
	White Page	A white page means this section is disabled and is not included when you generate the document.
	Template	A yellow template page gives specific format and styles that are included when you generate a document.
	Global Attribute	A global (red) attribute means it is passed down to every section below it (all children sections).
	Local Attribute	A local (green) attribute means it is only used by the section in which it appears.
	Grey Attribute	A grey attribute is used for comments.

Icon	Name	Description
	Control Icon	A Control type defines the functionality that you can give to Attributes. Each Control type has a different functionality.

Advanced editing mode allows you to add the following:

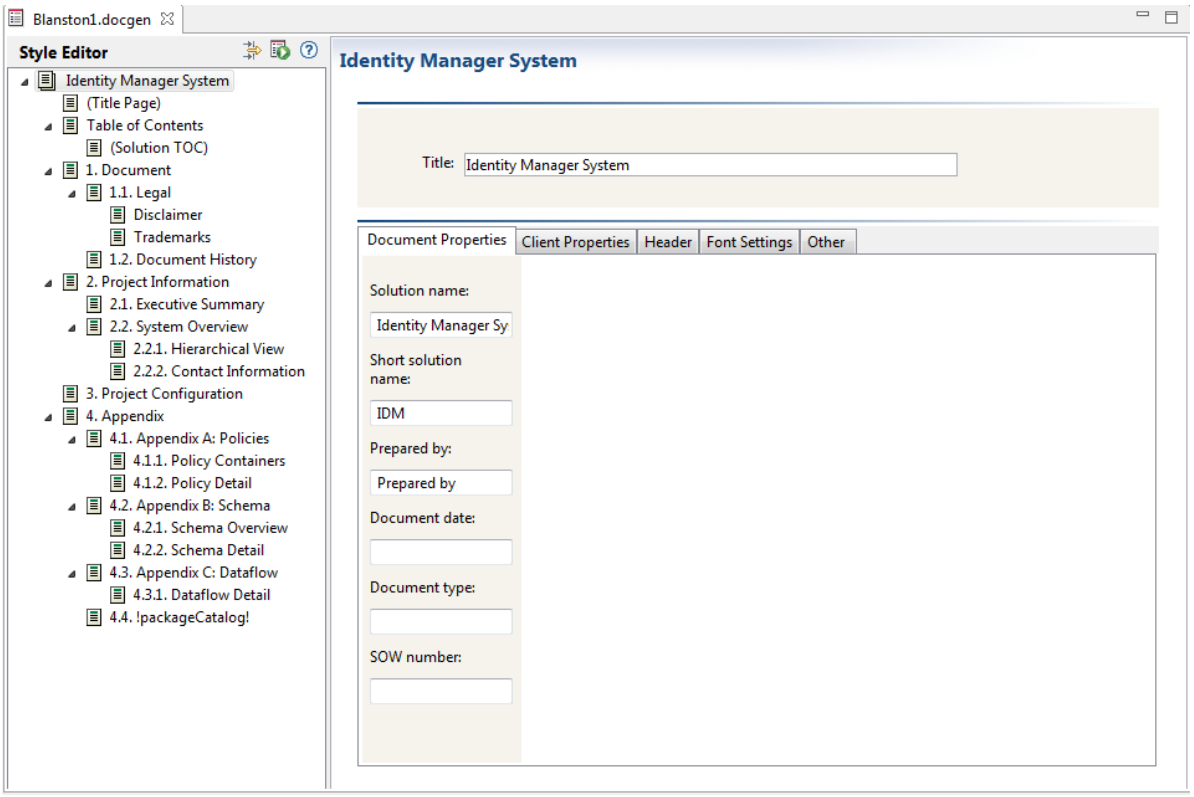
- “Sections” on page 335
- “Viewing or Editing Properties of a Section” on page 337
- “Templates” on page 338
- “Attributes” on page 339
- “Controls and Parameters” on page 339

You can have multiple sections in a document, but only one template per section. The template defines the section’s layout; however, you can use the default template for newly created sections. You can also have multiple attributes defining how the section looks, as well as multiple controls. You use parameters (such as names and values) to specify options for a Control type. A Parameter is a general name for a child item of a Control. The name of the Control denotes the type of control and what you can perform by using that type.

Sections

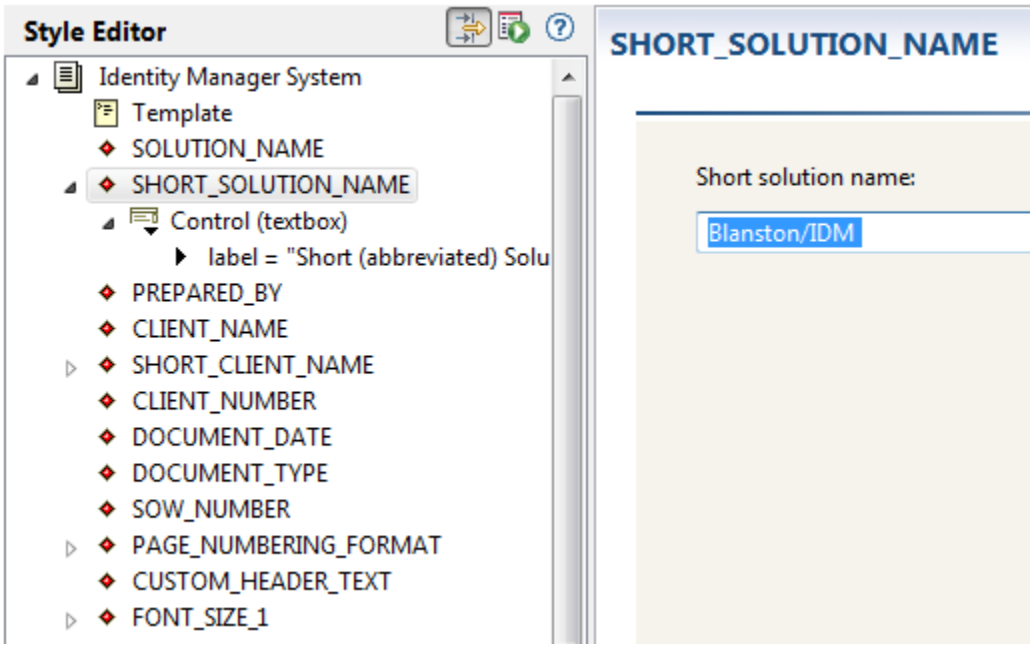
Sections are blocks of the documentation composed of attributes, parameters, templates, and controls through XSL programming. Section content includes a Title, Body, and children or subordinate information. The following example shows the Section Properties page of the OpenText Identity Manager System as seen through the simple edit mode.

Figure 12-3 *Section Properties Page*



The OpenText Identity Manager System section contains a section title (this can be changed), along with a number of tabs (attributes): **Document Properties**, **Client Properties**, **Header**, **Font Settings**, and **Other**. Each of these tabs contain fields that are editable; for instance, you can give the section title a different name. When you click the **Advanced Editing** icon, you see that the OpenText Identity Manager System section contains one template that includes several attributes, controls, and parameters underneath the heading.

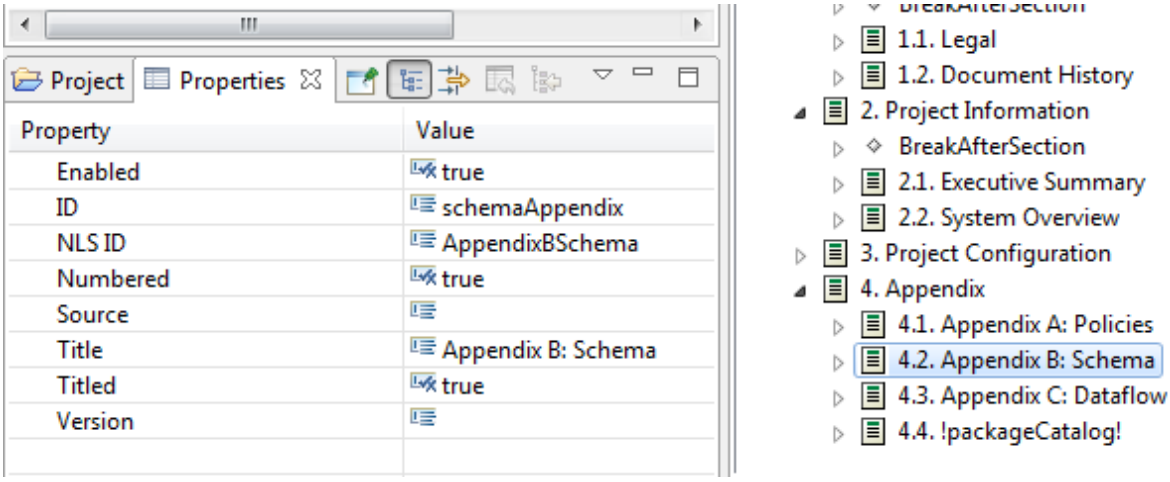
Figure 12-4 *Advanced Editing Mode*



Viewing or Editing Properties of a Section

If you click a section within the Style Editor and look in the Properties view (by default the lower left corner of Designer), you see the values associated with the selection. (If you do not see the Properties view by default, right-click a section and select **Show Properties View**.)

Figure 12-5 *The Properties View of the Appendix B: Schema*



These values are edited in the Properties view. The Values for the section heading are listed in the following table:

Table 12-3 *Values of a Section*

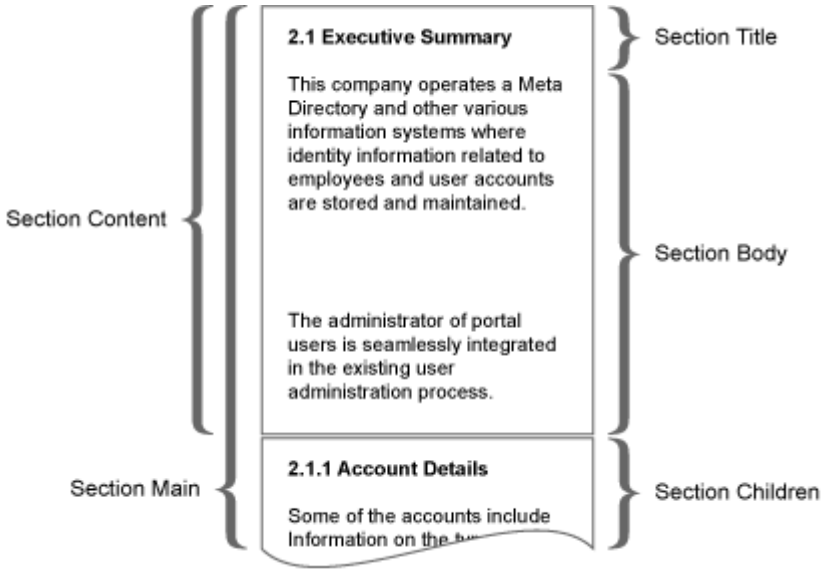
Property Name	Description
Enabled	(True/False) Indicates whether this section is enabled. You can change this setting by using the section Style Editor, which is the editor shown to the right of the hierarchical view.
ID	Used for reference. Most the time, ID is left empty. However, you can specify an ID for convenience in finding this section during the transformation process.
NLS ID	Used for reference. Most the time, NLS ID is left empty.
Numbered	(True/False) Indicates whether this section should be included in the numbering and placed in the Table of Contents.
Source	Data source used to transform the template. For example, designer, style, and none.
Title	The value to be displayed as the title. You can change this setting by using the section Style editor, which is the editor shown to the right of the hierarchical view.
Titled	(True/False) Indicates whether the title value should be shown in the generated document. Otherwise, it is used only in the GUI for context.
Version	The version of the section.

NOTE: Values change, depending on what you select under the Style Editor view. For example, an attribute shows different properties than a section or a template.

Templates

A template is the XML source that defines the overall layout of a generated page. For instance, the Title Page contains a template, as well as a number of headings. The following figure illustrates the parts of the Executive Summary template. For more information about templates, see "A Walk-through Tutorial" on page 340.

Figure 12-6 *Parts of the Executive Summary Template*



Attributes

Attributes are the child elements of a section. For example, clicking the Advanced Editor mode while selecting the section title OpenText Identity Manager System reveals the following attributes in red (global), green (local), and grey (comment):

Controls and Parameters

You can add parameters to control the appearance of a style. For example, in the Advanced Editing mode, the structure of the Short (abbreviated) Solution Name entry is a global attribute that contains a control and a label, and the control type known as textbox allows anyone to type a name that appears in the generated document. Use the Properties view to edit controls.

Designer's supported parameters or values for controls include the following:

Table 12-4 *Supported Parameters*

Control	Parameter or Value	Description
Table	columns="3" header="date" width="30" label="show this" addrows removerows	Number of columns to show in the control. Column header text. Column width for each column. Explains what you see in this control. Displays a button to perform this function. Displays a button to perform this function.
File	extensions=".jpg;*.gif" label="show this"	Supported extensions separated by a semicolon (`*.jpg;*.jpeg;*.gif`). (One file only.) The label explains what you can do with this control.

Control	Parameter or Value	Description
Select (OpenText Identity Manager System/font Settings)	option="font 1" option="font 2" label="show this"	Parameters allow font point selection, such as option = "20pt" and option = "24pt" The label indicates what you can do with this control.
Checkbox	label="show this"	The label explains what you can do with this control. It includes a check box.
Textbox or Textarea	label="show this"	The label explains what you can do with the text box or text area control. You edit these controls through the Properties view.
Comment	label="show this"	Allows you to add comments to help users. You edit this control through the Properties view.

A Walk-through Tutorial

Now that you better understand what components are necessary in order to add advanced functionality to your template, use this section to create a new section, add an attribute, and view the source.

- "Creating a New Section in a Style" on page 340
- "Adding an Attribute to a Style" on page 340
- "Enabling Documents to Recognize Your Additions" on page 341
- "Viewing the Source" on page 346

Creating a New Section in a Style

To insert an additional section into the Style Editor:

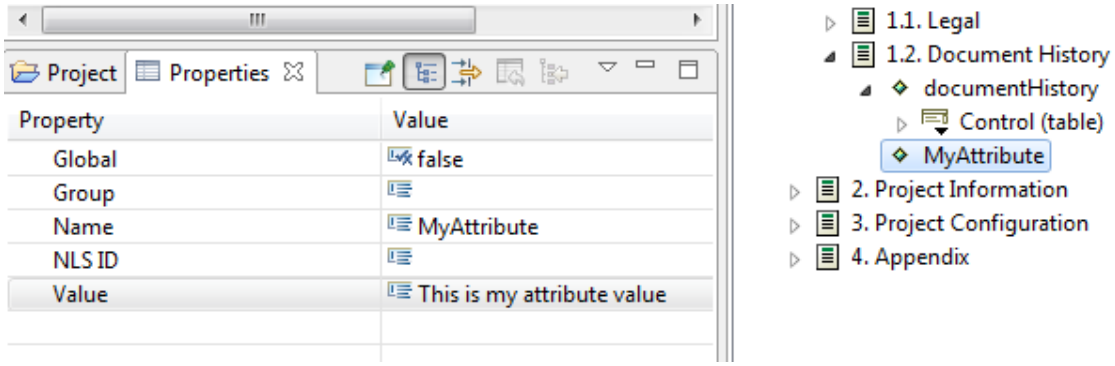
1. Create or open an existing `.docgen` file in the Style Editor.
2. Click the **Enable Advanced Editing** icon.

   The tree view of the document outline expands to include additional objects, (such as attributes, template, and parameters).
3. Right-click the parent section where you want to add your new section, then select **New Child > Section**. Specify a new section name; for this example, call it "My Section."
4. (Optional) Reorder the section by dragging the section object to a different location in the navigation tree. You can also copy and paste within this style or other styles.
5. Click the **Save** icon, then continue with "Adding an Attribute to a Style" on page 340.

Adding an Attribute to a Style

1. Right-click a section under the Style Editor view. Select **New Child > Attribute**.
2. Specify the attribute name in the Attribute Name window. For example, MyAttribute with no spaces. Click **OK**.

3 Specify a value under the Value property in the Properties view. For example, **This is my attribute value**.



The Properties view shows the following values for attributes (attributes are defined through controls and parameters):

- **Global:** Passed down to subsections.
- **Group:** Used to group attributes together. These appear as part of a tab in Style Editor's simple mode.
- **Name:** The attribute's name.
- **NLS ID:** The attribute's NLS identification.
- **Value:** The attribute's value.

You can also show your attribute with another control type. The following example first creates a control, then changes the control type from check box (the default) to something else, such as a text area.

1 Right-click **MyAttribute** and select **New Child > Control**.

2 With your cursor on the control you just created, change the control type value to **Text Area** in the Properties view.

3 Click your section to see the changes take place.

Enabling Documents to Recognize Your Additions

After you have added attributes, your generated document doesn't include information from these attributes until you do one of the following:

- Make sure your section is a leaf section (does not contain child sections).
- Create a template that uses the attribute explicitly. This is usually the preferred method because you can display the attribute values exactly the way you want.

Method 1: Set the Section Source to "Style"

If you create a section without a template (and the template is a leaf section containing no section children), the default template generates the attribute values with the document. You do not need to do anything. You can generate a document for just that section by right-clicking the section head and selecting **Generate Documentation for This Section**. Or you can click the **Generate Document** icon at the top to generate the whole document.

NOTE: Text boxes, text areas, and tables are the only attributes that are generated through the default template (check boxes, selects, and comments are not generated).

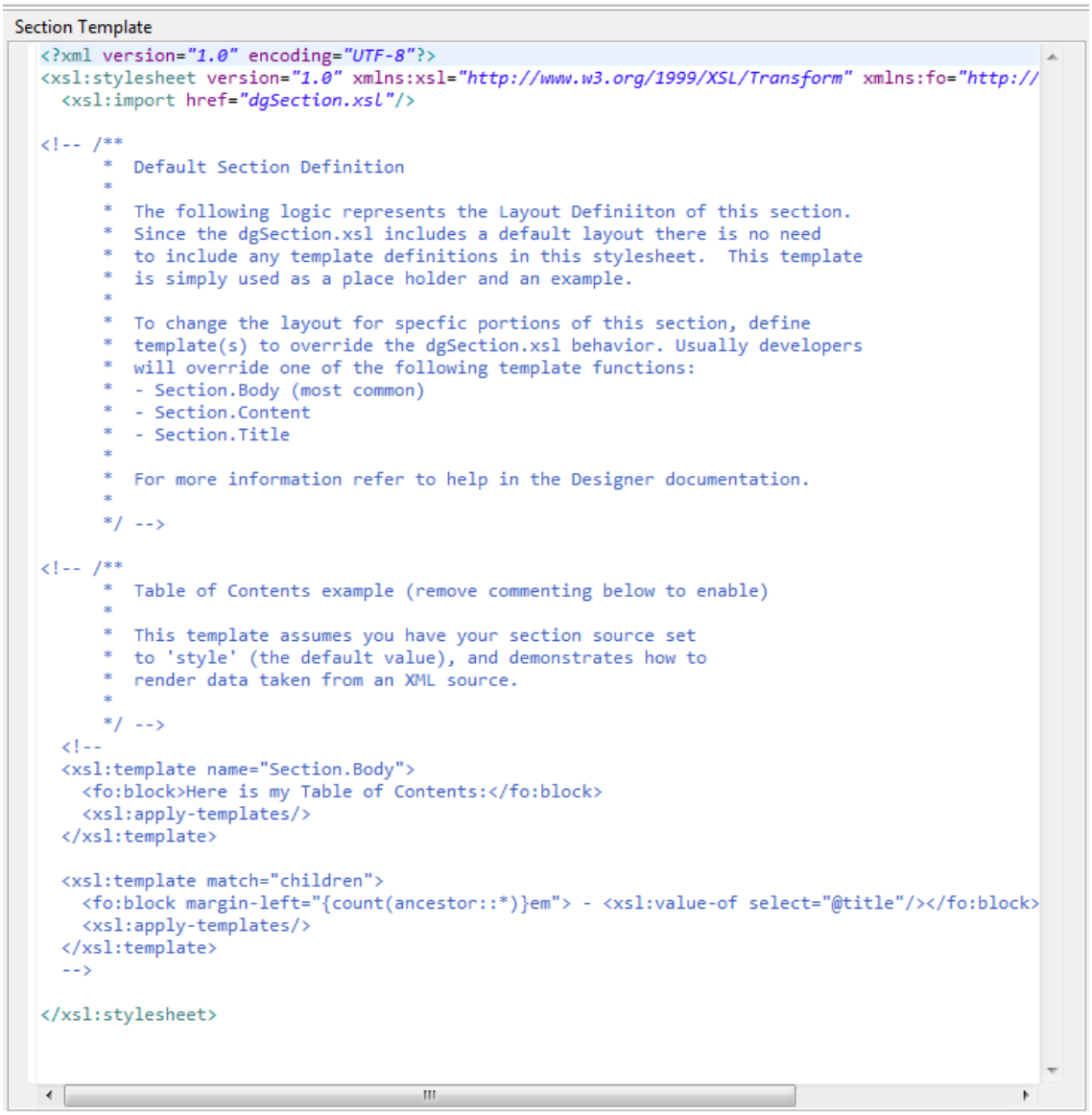
Method 2: Add a Template for a Custom Layout

Complete the following tasks:

- "Creating a Template" on page 342
- "Creating Another Section and Template" on page 344

Creating a Template

1. Right-click your newly created section.
2. Select **New Child > Template**.



The template has some default content that consists mostly of comments, which helps you get started on your first template. This is shown in the next task; for now, replace the comments in this template with the following XSL commands:

```
<xsl:param name="MyAttribute"/>

<xsl:template name="Section.Body">
  MyAttribute:
  <xsl:call-template name="Format.OutputTextArea">
     <xsl:with-param name="value" select="$MyAttribute"/>
  </xsl:call-template>
</xsl:template>
```

There is a Format.OutputTextArea call in the XSL that is a helper function included with the Document Generator Core Support Templates. Because HTML code is allowed in text areas, this ensures that it's interpreted and escaped properly. If you want to see the core XSL library calls for documentation generation, see "Document Generator Core Support Templates" on page 581.

Your template should look like this:

Figure 12-7 *Example Template*

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1
  <xsl:import href="dgSection.xsl"/>

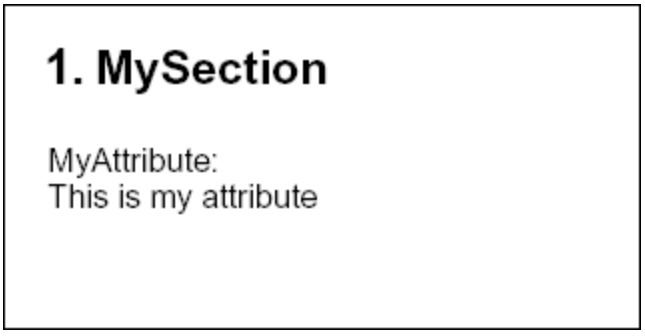
  <xsl:param name="MyAttribute"/>

  <xsl:template name="Section.Body">
      MyAttribute:
      <xsl:call-template name="Format.OutputTextArea">
        <xsl:with-param name="value" select="$MyAttribute"/>
      </xsl:call-template>
  </xsl:template>

</xsl:stylesheet>
```

Generating a document for this section should give you something like this:

Figure 12-8 *Sample Section*



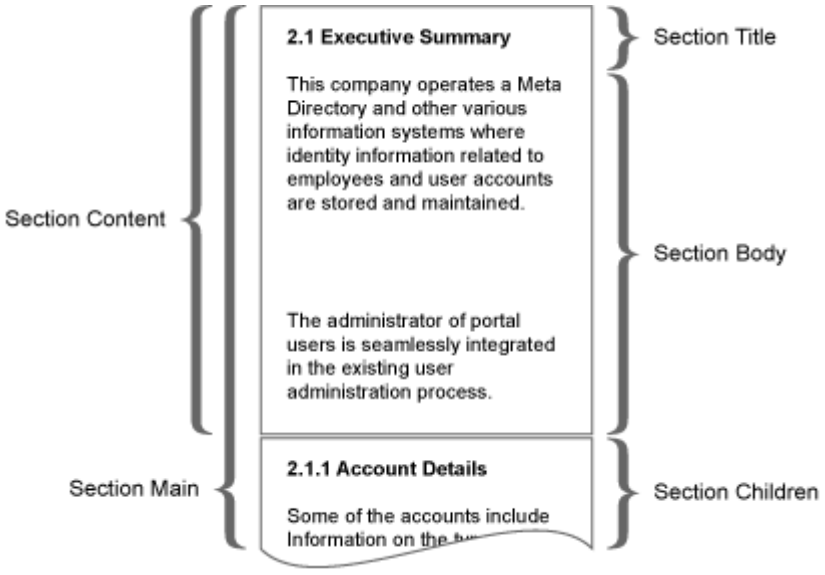
Creating Another Section and Template

1. Right-click your newly created section and select **New Child > Section**. In the Section Name window, name the new section Table of Contents and click **OK**.
2. Right-click this new section and select **New Child > Template**. Carefully read through the comments in this template. These details are important.

   When you create a custom section, you are inserting some information into the document. As the comments mention, developers usually override one of the following template functions:

   - Section Body (most common)
   - Section Content
   - Section Title

   The following image illustrates which section is being defined. As a developer of the style, you write this template to overwrite the area that is of interest.



   For this example, you should overwrite the Section.Body because you don't want to change the default behavior of the Title, nor do you want to change the way other sections are related to this one. (You can use the hierarchical view to control this with the default template if necessary).
3. To overwrite the Section Body, simply uncomment the sample function that is shown in the default template, as shown below:

Section Template

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
  <xsl:import href="dgSection.xsl"/>

<!-- /**
      *  Table of Contents example (remove commenting below
      *
      *  This template assumes you have your section source
      *  to 'style' (the default value), and demonstrates h
      *  render data taken from an XML source.
      *
      */ -->
  <xsl:template name="Section.Body">
    <fo:block>Here is my Table of Contents:</fo:block>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="children">
    <fo:block margin-left="{count(ancestor::*)}em"> - <xsl:
    <xsl:apply-templates/>
  </xsl:template>

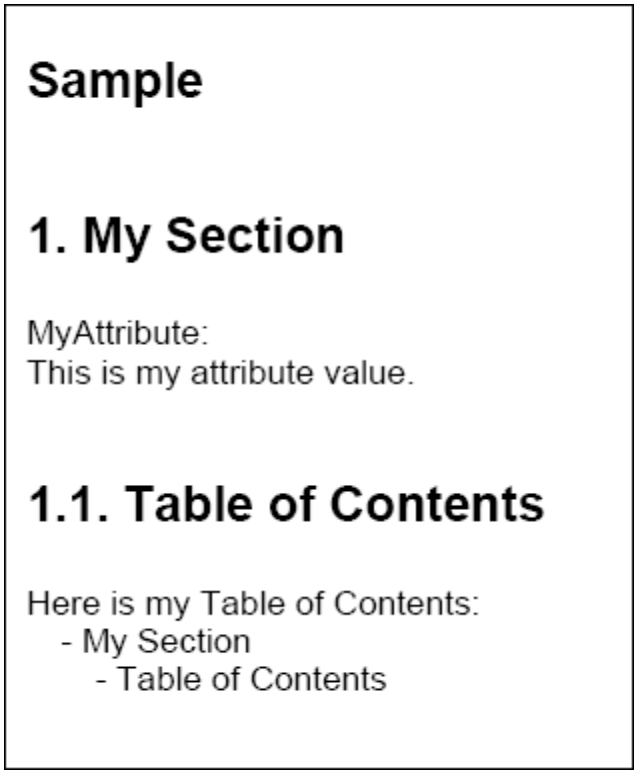
</xsl:stylesheet>
```

If you render your document at this point, you get no content in your Table of Contents (other than the surrounding text). This is because this template assumes that the style source has been specified for this section. To specify the source:

4 Click the Table of Contents section.

5 From the Properties view, set the source to **style**.

6 Right-click **MySection** and select **Generate Documentation for This Section**.



Viewing the Source

When your generate your document, you'll notice there is an **Output XML Source Files** option. Click the box next to this option to turn it on. You'll see `.xml` source files appear where you are generating the document. These source files are the XML data that is used in your template when you set the source (for example, to "style"). Designer includes the following sources:

Table 12-5 *Sources*

Source Key	Description
none	An empty source, used when no source is specified or when "none" is specified.
style	The XML source of the style, used to build things like the Table of Contents.
designer	A source that has been defined by an extension point for the Designer model. This contains all information about the configuration of your Designer project.

Selecting a Language for Generated Documents

You can select the language you want to print the document in.

1 Click **Window > Preference > Designer for IDM > Document Generation**. Under the **Document Language** heading, select the language you want to use for document generation.

Current languages include:

- Chinese Simplified
- Chinese Traditional
- Dutch
- English (default)
- French
- German
- Italian
- Japanese
- Portuguese Brazil
- Spanish

2. After you select a language, click **Apply**.
3. Click **OK** to close the Preference page.

Double-Byte Font Support

Designer has double-byte font support for the Document Generation feature. If you select a language that uses double-byte characters, such as Chinese Simplified, Chinese Traditional, and Japanese, Designer automatically installs the Proportional Mincho font. You can change this as necessary. A good font that covers both proportional spacing and double-byte support is Arial Unicode MS.

For English and other languages, the default font is Arial.

To add a font for your specific language:

1. Click **Window > Preference** then expand **NetIQ > OpenText Identity Manager** and select **Document Generation**. Under **Document Appearance**, select the font you want to use.

   To change the font on a Windows workstation, you must first copy the font file from the `C:\Windows\Fonts` directory to another directory. You can then use the **Browse** icon to select the font.

   To change the font on a Linux workstation, browse to the `usr/share/fonts/truetype` directory, or to another directory containing the fonts you want.

2. Click the **Browse** icon to bring up the Open window, change to the directory where you placed the font, then click **Open**.

   You can also type the directory and font file name into **Font Settings** field, or use the drop-down menu to select a font that you have previously selected.

3. Click **Apply**, then click **OK**.

Using the above steps globally changes the font in the generated document, and also adds double-byte font support for your selections.

13 Using Entitlements

OpenText Identity Manager allows you to synchronize data between managed systems. Entitlements allow you to set up criteria for a person or group that, when met, initiate an event to grant or revoke access to business resources within the managed system.

You can think of an entitlement as a permission slip. For example, if you want a new employee to be given an Active Directory account when he is added to your Human Resource system, he must have a permission slip, or entitlement, for the Active Directory account. If the user doesn't have the permission slip, he doesn't receive the account. This gives you one more level of control and automation for granting and revoking resources.

Use Designer to create entitlements and deploy them into existing OpenText Identity Manager drivers. Designer allows you to create entitlements through the Entitlement Wizard, which gives you a graphical interface where you can create the entitlement step by step. Because of this graphical interface, we recommend using Designer for creating and editing entitlements.

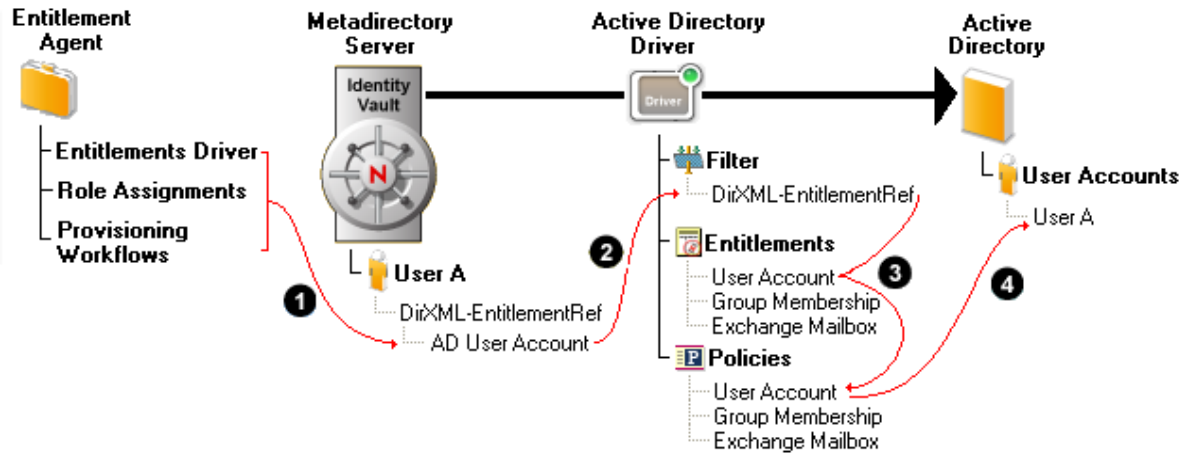
There are four aspects to making entitlements work effectively: design, creation, editing, and management.

- "How Entitlements Work" on page 349
- "Designing Entitlements" on page 351
- "Creating Entitlements through the Entitlement Wizard" on page 354
- "Editing and Viewing Entitlements" on page 368
- "Managing Entitlements" on page 372

How Entitlements Work

The following diagram shows the basic entitlement process.

Figure 13-1 *Basic Overview of Entitlements*



1. An entitlement agent grants an entitlement to a user. There are three ways that entitlements are granted to a user:

   - **Role-Based Entitlements:** The Entitlements Service driver grants the entitlement based on criteria that places the user in a particular role (or group). This criteria can be based on any event that occurs in the Identity Vault. For example, adding a new employee in an HR system causes a User object to be created in the Identity Vault. Creation of the new User object is the criterion that causes the Entitlements Service driver to grant the Active Directory User Account entitlement to the user.

     To create role-based entitlements in Designer, see "Creating Entitlements through the Entitlement Wizard" on page 354.

   - **User Application Role Based Provisioning:** The user receives a role assignment through the User Application. The User Application's Role Service driver grants the user any entitlements associated with the new role. For example, a user is assigned an Accountant role that requires access to the Accounting group in Active Directory. The Role Service driver grants the Active Directory Group Membership entitlement to the user.

     To create entitlements for role based provisioning, use the Role editor. See *Specifying Entitlements* in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Administrator's Guide to Designing the Identity Applications*.

   - **User Application Workflow-Based Provisioning** A provisioning workflow grants the entitlement to the user. For example, a new employee is added to the HR system, which causes a User object to be created in the Identity Vault. Creation of the new User object initiates a workflow that grants the Active Directory User Account entitlement to the user.

     Creating entitlements to use with workflow-based provisioning is an involved process. To get you started, see"Configuring Provisioning Request Definitions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Administrator's Guide to Designing the Identity Applications*.

2. When an entitlement is added to or removed from a user's DirXML-EntitlementRef attribute, any entitlement-enabled drivers begin to process the event. To monitor users for entitlement changes, drivers must have the DirXML-EntitlementRef attribute added to their Subscriber channel filter.

3. The driver processes the entitlement event against the Subscriber channel policies. If the entitlement event is for an entitlement that applies to the driver, the policies are processed. Otherwise, no processing occurs. In Figure 13-1, the Grant User Account policy is processed because:
   a. The Active Directory User Account entitlement was added to the user's DirXML-EntitlementRef attribute.
   b. The User Account entitlement is defined on the Active Directory driver.

   Likewise, if the Active Directory User Account entitlement is later removed from the user's DirXML-EntitlementRef attribute, the Revoke User Account policy is processed.
4. The policies trigger the granting or revoking of access to the entitled resource. In Figure 13-1, the Grant User Account policy triggers the creation of a user account in Active Directory.

Designing Entitlements

You must know beforehand what you want to accomplish with entitlements. Entitlements work from the functionality you build into OpenText Identity Manager drivers through policies. These driver policies implement rules and process the events between the Identity Vault and the managed system. If the policies in the OpenText Identity Manager driver do not specify what you want to do, entitlements cannot work. For example, if you don't specify the action section of the Check User Modify for Group Membership rule in the Command policy, attempts to grant or revoke a group membership entitlement are ignored.

When you know what you want to accomplish with OpenText Identity Manager, you can correctly design granting and revoking capabilities for any managed system resources. The following four-step procedure can help you plan to create and use entitlements:

1. Know what you want to accomplish in your business situation. You can design and implement many business solutions through OpenText Identity Manager, but you need to know what you want to do before implementing something that isn't defined. Make a numbered list of what you want to do.
2. Define an entitlement that represents one item from your numbered list. You can create valueless and valued entitlements. Valued entitlements can get their values from an external query, they can be administrator-defined, or they can be free-form. There are examples in "Creating Entitlements through the Entitlement Wizard" on page 354.
3. Add policies to the OpenText Identity Manager Driver to implement the designed entitlement. To create a policy for an OpenText Identity Manager driver, you need to be conversant with XSLT or DirXML script to define the way the managed system handles and receives information, and the way OpenText eDirectory stores information. Unless you are a good DirXML programmer, this is a job for consultants.
4. Set up a managing agent to grant or revoke the entitlement. If you want an automated process, use Role-Based Entitlements; if you want a manual process, use the User Application's workflow-based provisioning feature. See "Understanding Entitlements" in the *OpenText™ Identity Manager Entitlements Guide*. As you plan your entitlements, use the following sections for more information.

- "Terminology" on page 352
- "Entitlement Prerequisites" on page 352

- "OpenText Identity Manager Drivers with Preconfigurations that Support Entitlements" on page 352
- "Enabling Entitlements on OpenText Identity Manager Drivers" on page 353

Terminology

Following are some terms that are used throughout this section.

Entitlement: An Identity Vault object that represents a business resource in a managed system.

Entitlement Service driver: Grants and revokes entitlements. For Role-Based Entitlements, the agent is the Entitlements Service driver, which must be initiated for entitlements to work.

Grant or revoke: Granting or revoking an entitlement is controlled by Global Configuration Variables (GCVs) on an OpenText Identity Manager driver.

Entitlement consumer: Anything that uses entitlement-related information. Entitlement consumers include iManager, the User Application, and Identity Manager policies.

Entitlement Prerequisites

- ❒ eDirectory 8.7.3 or eDirectory 8.8 with the latest Support Pack
- ❒ Identity Manager 3 or later
- ❒ An Entitlements Service driver

  You must have an Entitlements Service driver in each driver set where you want to use entitlements. This requires a very simple, two-step setup for each driver set. To do this, see "Creating Entitlements" in the *OpenText™ Identity Manager Entitlements Guide*.
- ❒ A driver configuration that supports entitlements

  Before you can use entitlements with a managed system, do one of the following:

  - Import the OpenText Identity Manager driver configuration for the driver and specify that the driver has entitlements enabled.
  - Enable your driver to support entitlements. To do this:

    1. Create entitlements using Designer.
    2. Add the DirXML-EntitlementRef attribute to your driver filter as described in "Enabling Entitlements on OpenText Identity Manager Drivers" on page 353.
    3. Write policies to implement the entitlements you create in Step 1 under "Designing Entitlements" on page 351.

OpenText Identity Manager Drivers with Preconfigurations that Support Entitlements

The following drivers include configuration files that already contain entitlements and the policies required to implement the entitlements. These entitlements support the most common scenarios: granting and revoking user accounts, groups, and e-mail distribution lists.

- Active Directory: Grant and revoke accounts, group membership, Exchange Mailbox
- GroupWise 2014: Grant and revoke accounts, grant and revoke members of distribution lists

- LDAP: Grant and revoke user accounts
- Linux and UNIX: Grant and revoke accounts
- Lotus Notes: Grant and revoke user accounts and group memberships
- RACF: Grant and revoke group accounts and group memberships

These are example entitlements and policies that you can use as is if they meet your needs. If not, you can modify them to meet your needs, or you can use them as examples as you implement additional entitlements.

Enabling Entitlements on OpenText Identity Manager Drivers

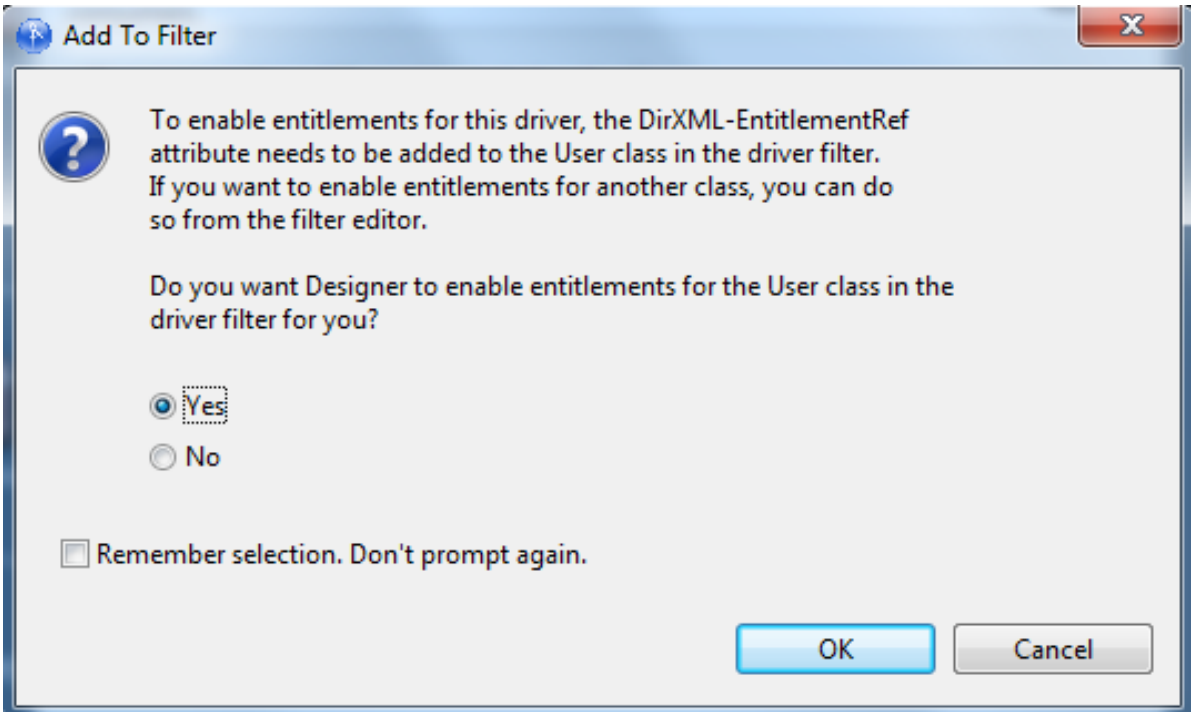
Before you can use entitlements, you must first ensure that your driver has entitlements enabled. You can do this through the Entitlements Wizard as you finish creating entitlements; this applies to both preconfigured and non-preconfigured drivers.

However, if you want to use the preconfigured driver's entitlements and the infrastructure that supports them, you must enable entitlements when you initially create a driver in Designer or iManager; the preconfigured policies and rules that support the preconfigured entitlements cannot be added later without re-creating the driver. If you import a driver that has entitlements enabled into Designer from an Identity Vault, the imported driver also has entitlements enabled. If you deploy a driver that has entitlements enabled into an Identity Vault, the deployed driver also has entitlements enabled.

You can see if your preconfigured drivers have entitlements enabled by clicking the Outline view, then clicking the Subscriber channel of your selected driver. If entitlements are enabled, you should see the preconfigured entitlements appear under the Subscriber Channel. If entitlements do not appear under the Subscriber Channel in the Outline view, entitlements were not enabled when the driver was initially installed.

However, you can still use entitlements on preconfigured OpenText Identity Manager drivers that do not have entitlement preconfigurations enabled. To do this, run the Entitlement Wizard. The last page in the Entitlement Wizard asks if you want to add the DirXML-EntitlementRef attribute to the driver filter, with **Yes** selected. Click **OK**. However, because the policies and rules are not in place on the driver, you won't be able to use their preconfigured entitlements without adding those supporting policies and rules yourself.

Figure 13-2 *Enabling Entitlements*



You can also use entitlements on OpenText Identity Manager drivers that do not contain entitlement preconfigurations. To enable your driver to support entitlements, add the DirXML-EntitlementRef attribute to your driver filter. Run the Entitlement Wizard as described above to add the DirXML-EntitlementRef attribute to the driver filter.

Creating Entitlements through the Entitlement Wizard

Designer comes with an Entitlement Wizard. This wizard steps you through the creation of entitlements by asking a series of questions about how the entitlement will be used in the enterprise. Use one of the following methods to access the Entitlement Wizard:

To access the Entitlement Wizard from the Outline view:

1 Right-click a Driver object, then click **New > Entitlement**.

To access the Entitlement Wizard from the Modeler view:

1 Right-click the driver icon, then click **New > Entitlement**.

There are two types of entitlements that you can create: valueless and valued. The type you use depends on whether you need to pass additional information to the policies.

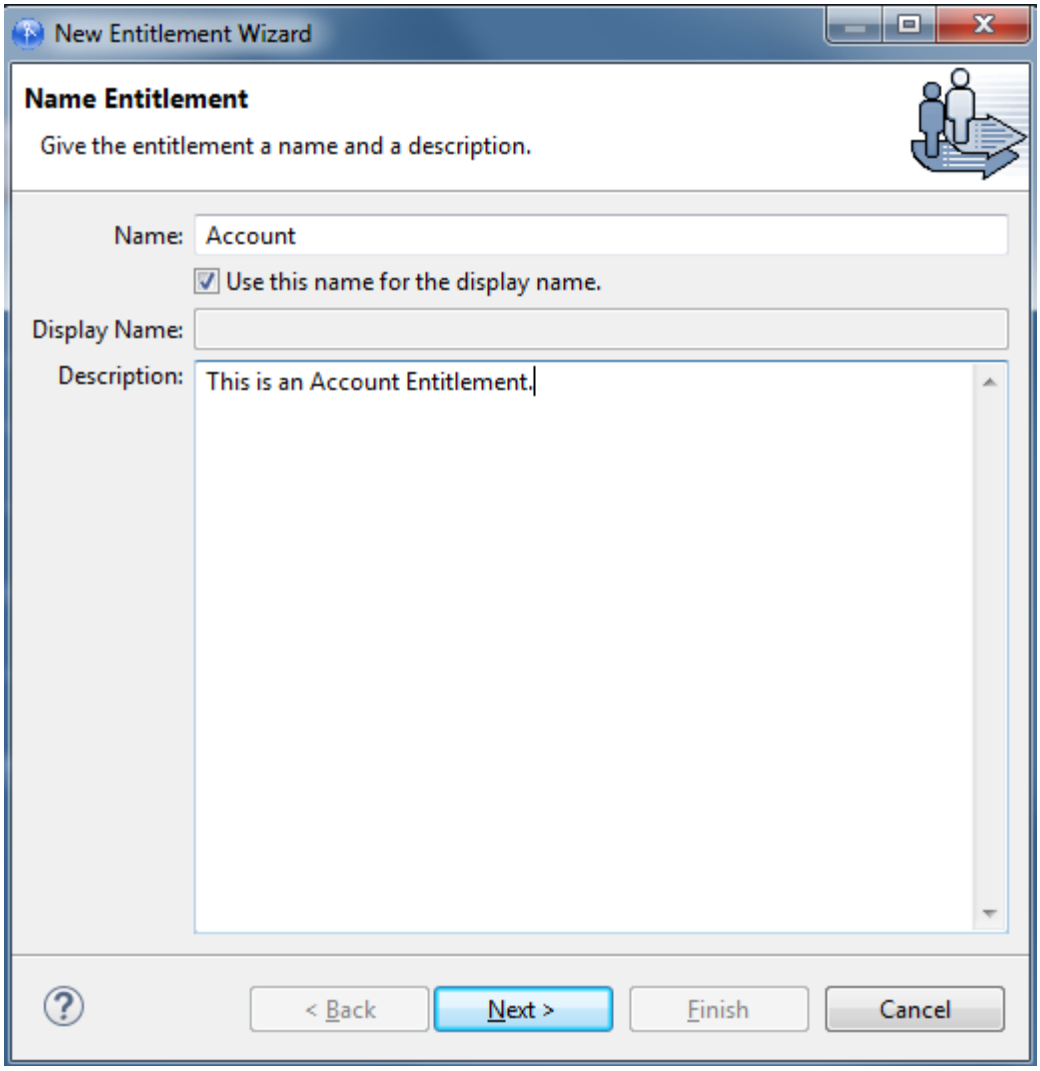
- “Valueless Entitlements” on page 355
- “Valued Entitlement that Queries an External Application” on page 357
- “Administrator-Defined Entitlements with Lists” on page 364
- “Administrator-Defined Entitlements without Lists” on page 366

Valueless Entitlements

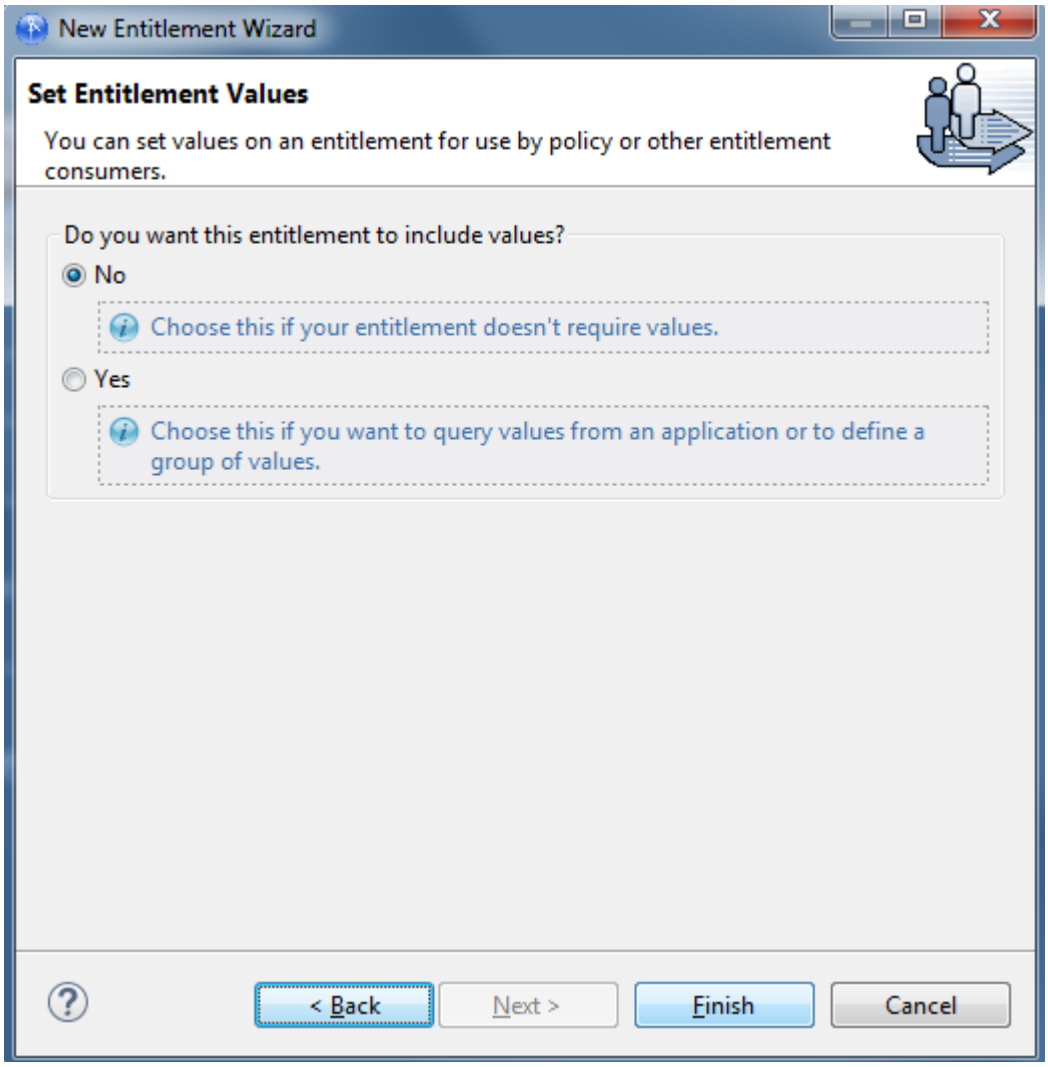
A valueless entitlement has no values to go with it. An example is the Account Entitlement for Active Directory, which is used to turn on account capabilities. You use valueless entitlements if you don't need to pass any extra information to driver policies.

To create a valueless entitlement:

1 Right-click the driver icon in the Outline view or in the Modeler view, then click **New > Entitlement**.



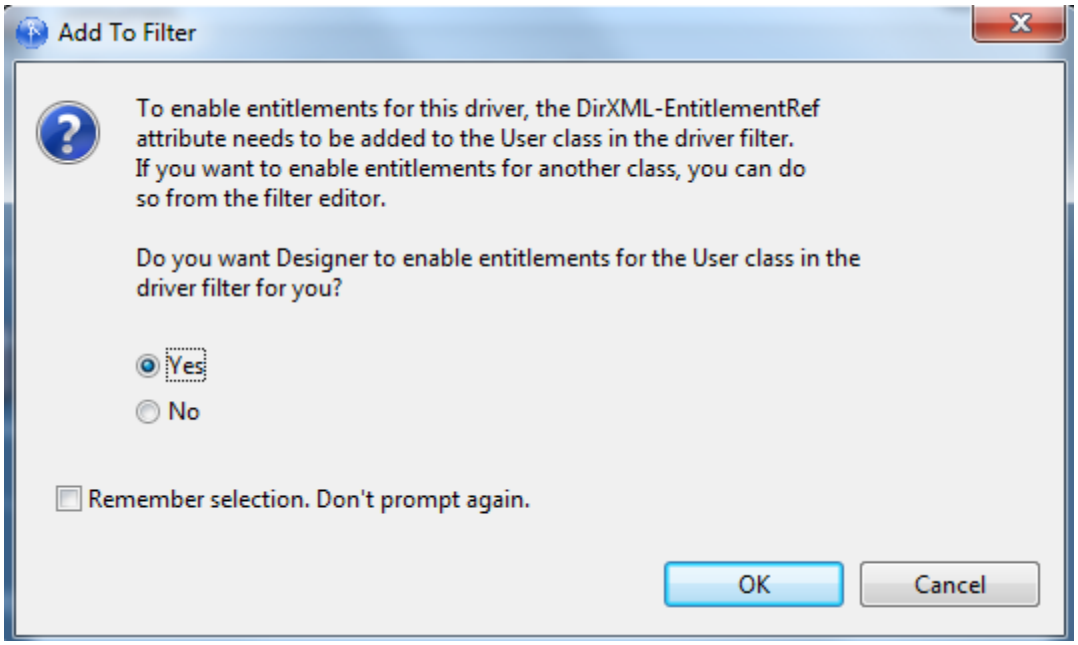
2 Type the name and description information. For this example, the entitlement is named Account, with a description of "This is an Account Entitlement." Click **Next**.



3 Because this first example is valueless, select **No** to **Do you want this entitlement to include values?**

4 Click **Finish**.

5 In the Add To Filter dialog box, answer **Yes** if you want the driver to listen for this entitlement. This enables entitlements for the driver.



The DirXML-EntitlementRef attribute allows the driver filter to listen for entitlement activities. This is necessary in order to use the entitlements you are creating.

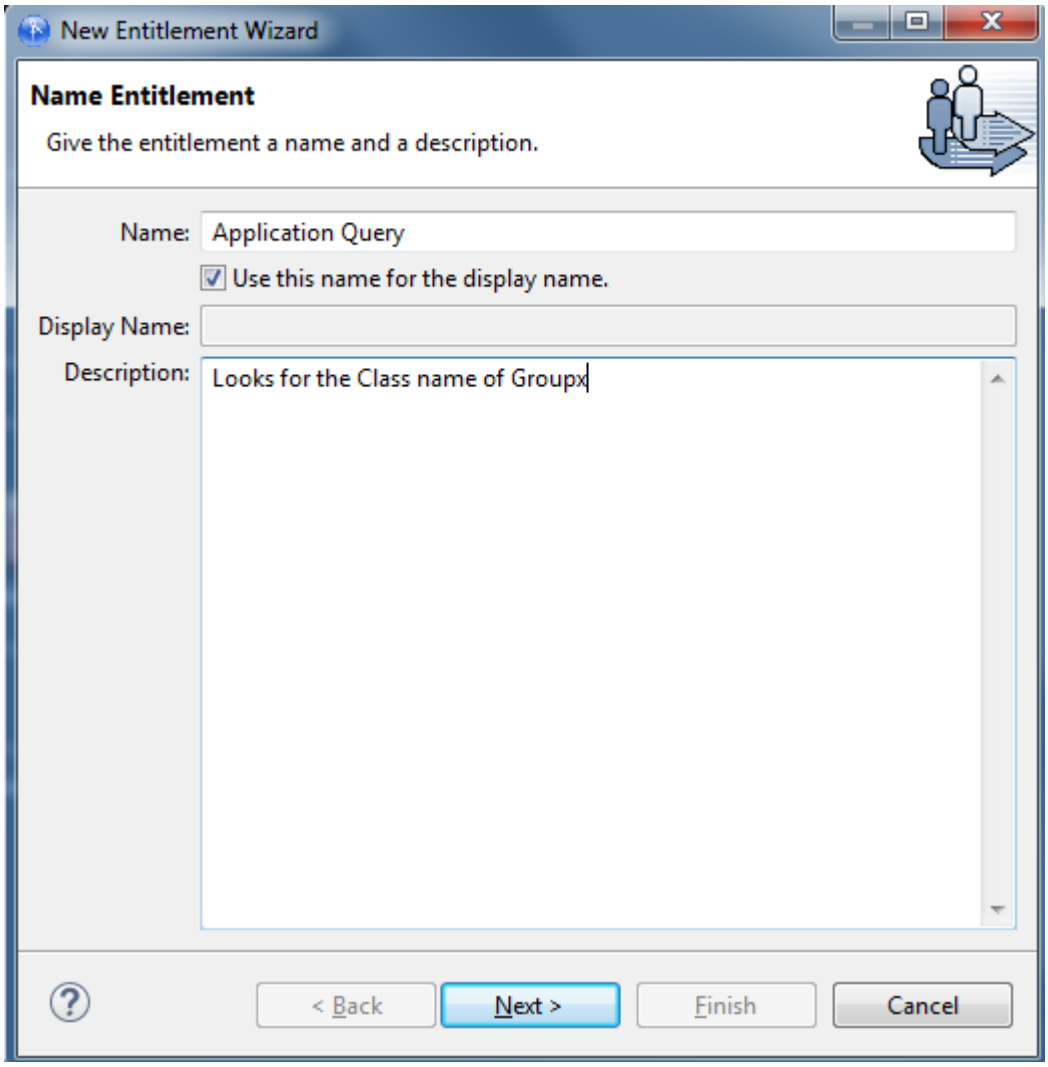
If you don't want to see the Add To Filter window on every entitlement you are creating for any driver in Designer, select **Remember selection - Don't prompt again**, then click **OK**. However, after the attribute is added to this driver filter, you won't see the Add To Filter window again.

If you have a file conflict, you are asked to save the editor's changes before continuing. After the editor is saved, the entitlement displays in the Modeler view.

Valued Entitlement that Queries an External Application

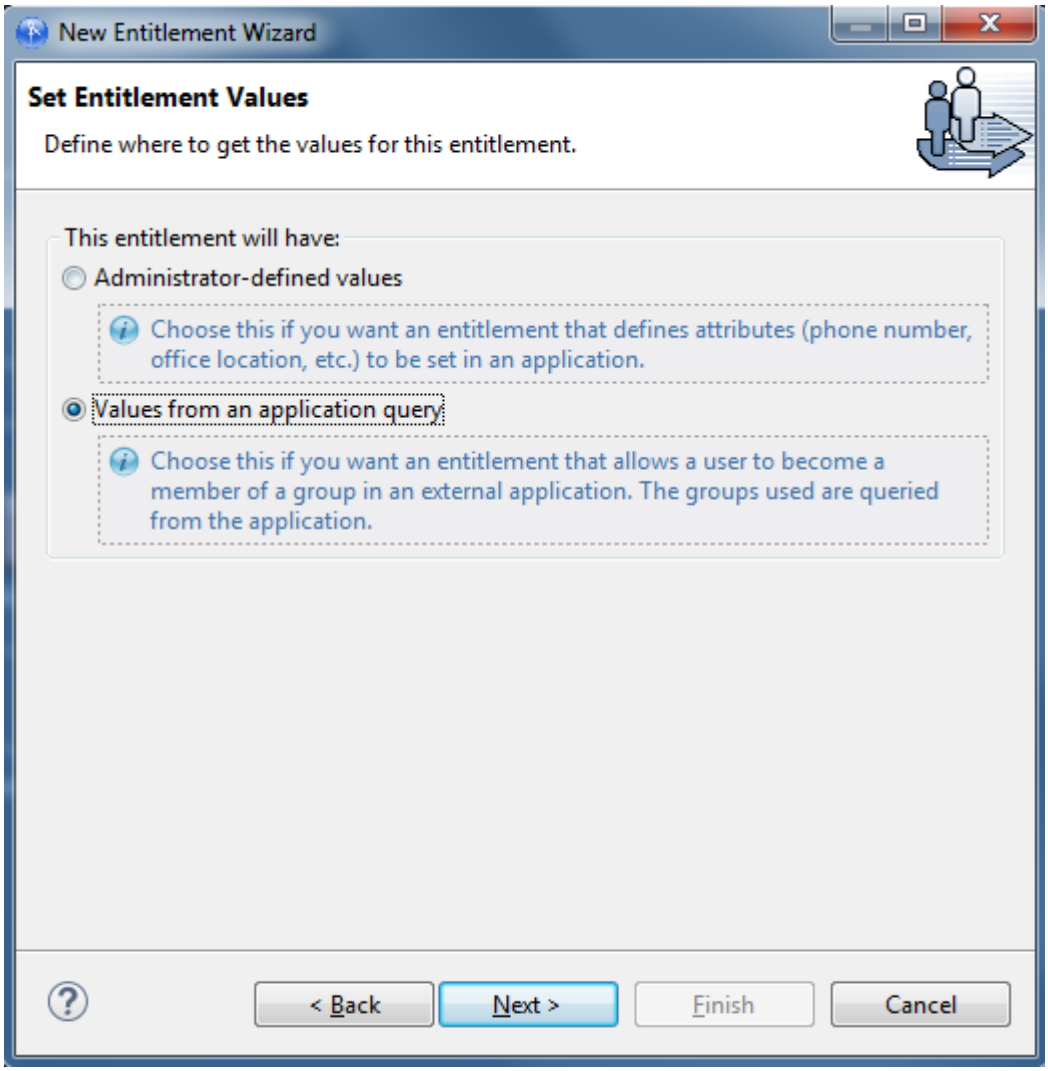
Values are a way of passing data that you might need to use in policies. Valued entitlements can get their values from an external query; they can be administrator-defined, or they can be free-form.

1. Right-click the driver icon in the Outline view or in the Modeler view, then click **New > Entitlement**.
2. Give the entitlement a name. This example uses Application Query, with the **Use Name for Display Name** option selected. In the Description box, type `Looks for the Class name of Groupx`, then click **Next**.



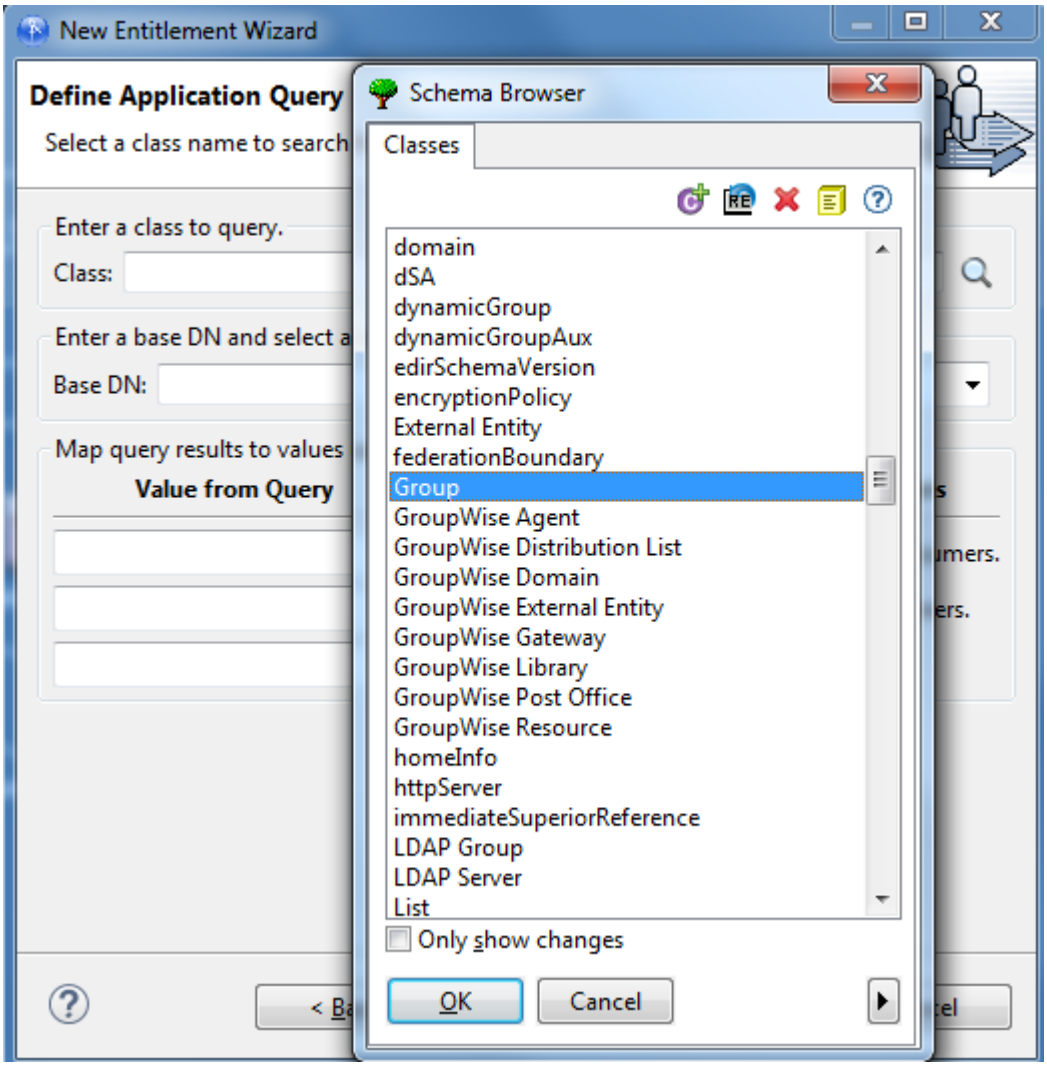
3 On the Set Entitlements Values page, select **Yes** so you can query values from an application or define a group of values, then click **Next**.

4 The next Set Entitlements Values page allows you to define where you get the values for this entitlement. Valued entitlements can get their values from an external query, or they can be administrator-defined. For this example, select the **Values from an application query** option, then click **Next**.

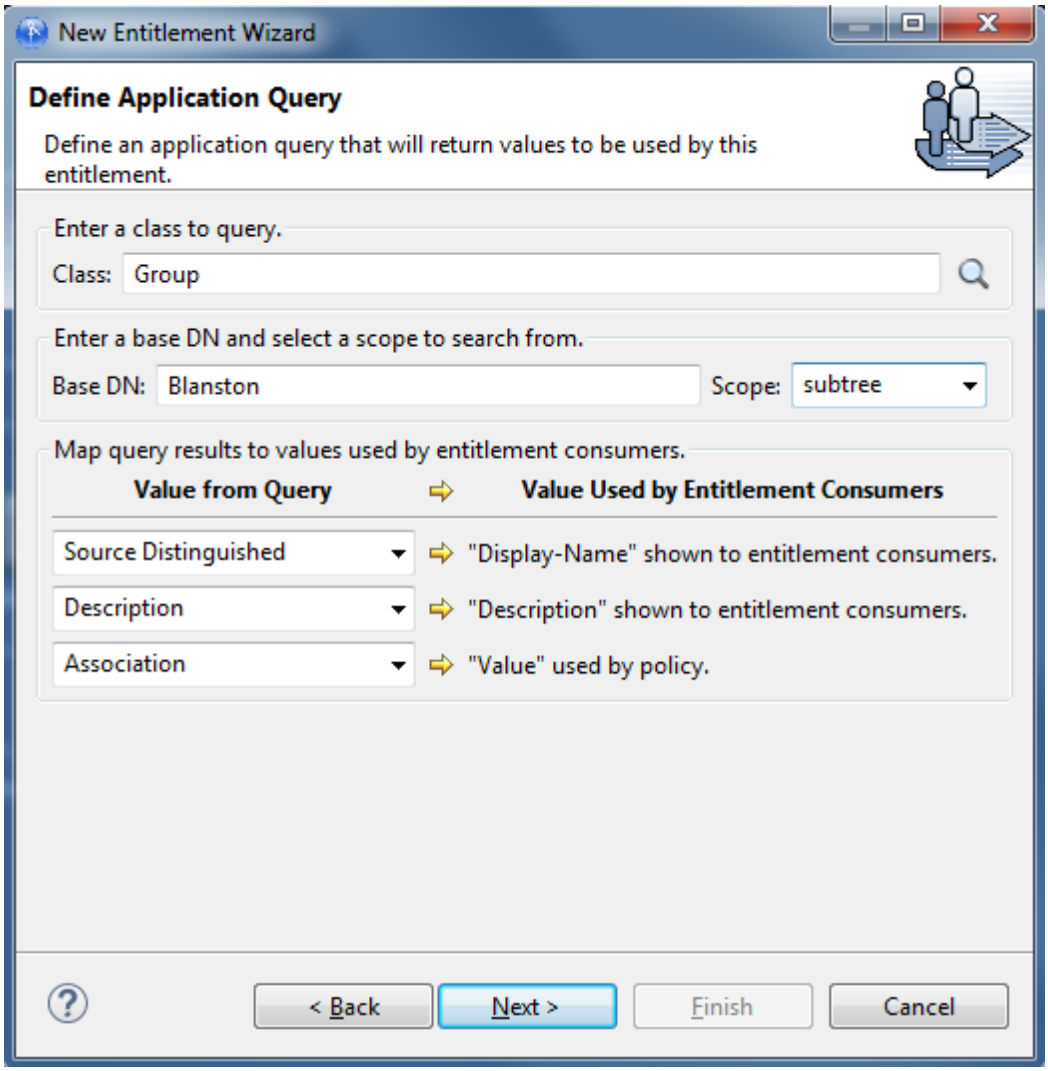


The Define Application Query window combines two steps: defining the query and mapping the query results.

5 To fill in the Class Query, click the **Schema Browser** icon on the right side of the Class entry.



6 The Schema Browser shows you the Classes in the eDirectory namespace that are available. If you know the name of the Class type you want to query, click to select a selection in the **Classes** tab, then start typing the Class name. The browser jumps to the alphabetical order of what you type. Select the Class name, then click **OK**.



7 Type the base distinguished name (DN) and the scope. For this example, select the Class Group, at the Base Distinguished Name of Blanston, with the Scope of subtree (choices are subtree, entry, and subordinates).

This example maps the query results from the managed system to certain values that entitlement consumers can use. At present, the consumers are iManager managing Role-Based Entitlement policies and the User Application managing workflow-based provisioning entitlements. The **Value From Query** information prepopulates the consumer's user interface with the following:

- **Display Name:** Defines the attribute that displays in the list of values. The example selects Source Distinguished Name for the display name. Click the drop-down button on the **Display Name shown to entitlement consumers** list to see a list of attributes associated with the class you selected through the Schema Browser. The list includes both the attributes and the inherited attributes for the selected class.
- **Description:** Defines the attribute that displays as a description for that value. For the description, select **Description** from the Value drop-down list to map the query results from the managed system to the entitlement.

- **Value:** Defines the attribute or token that is the actual value. The **Value** entry is not seen in the entitlement consumer, but it is the value that is assigned when the entitlement is granted or revoked. In this case, choose **Association**.

If you do not use the **Schema Browser** icon when selecting the class, you see only two selections in the **Value From Query** lists: **Association** and **Source Distinguished Name**. If these attributes suit your needs, use them. You can also type the attribute name into the text field. However, if you want to select the attributes from the lists, use the **Schema Browser** icon when selecting a class for the query. You see the attributes and inherited attributes for the selected class.

8 When the values are filled out, select **Next**.

9 In the Assign Multiple Values window, select **Yes** if you want the entitlements to be granted more than once and with different values. If you select **No**, the entitlement can only be granted once. For this example, click **Yes**, then click **Next**.

It makes sense to assign group entitlements with multiple values, but it does not make sense to assign an account entitlement more than once.

10 You are asked if this entitlement is intended to be used by Role-Based Entitlement policies through iManager. If you want this entitlement to be granted or revoked automatically, select **Yes** to the Role-Based Entitlements question, click **Next**, then continue with Step 11.

or

If you want the granting or revoking of this entitlement to be a manual process (approved by someone), select **No** to use the User Application, then skip to Step 12.

We recommend that you have only one agent control an entitlement. If multiple agents are in control, you have the following consequences:

- Whatever comes last controls the entitlement results
- Results are unpredictable
- Using both agents to control an entitlement is not supported by OpenText

11 (Conditional) If you select **Yes** to the Role-Based Entitlements question, you are asked if you want to use the Role-Based Entitlements priority to resolve any conflicts that might happen when this entitlement is assigned more than once with different values. You can resolve the conflict by either using Role-Based Entitlements priority, or by merging the values.

Merging the values merges the entitlements of all involved Role-Based Entitlement policies, so if one policy revokes an entitlement but another policy grants an entitlement, the entitlement is eventually granted. Solving conflicts by priority works if you need to ensure that only one policy is applied to this entitlement at any time. This example uses priority.

12 Click **Finish**.

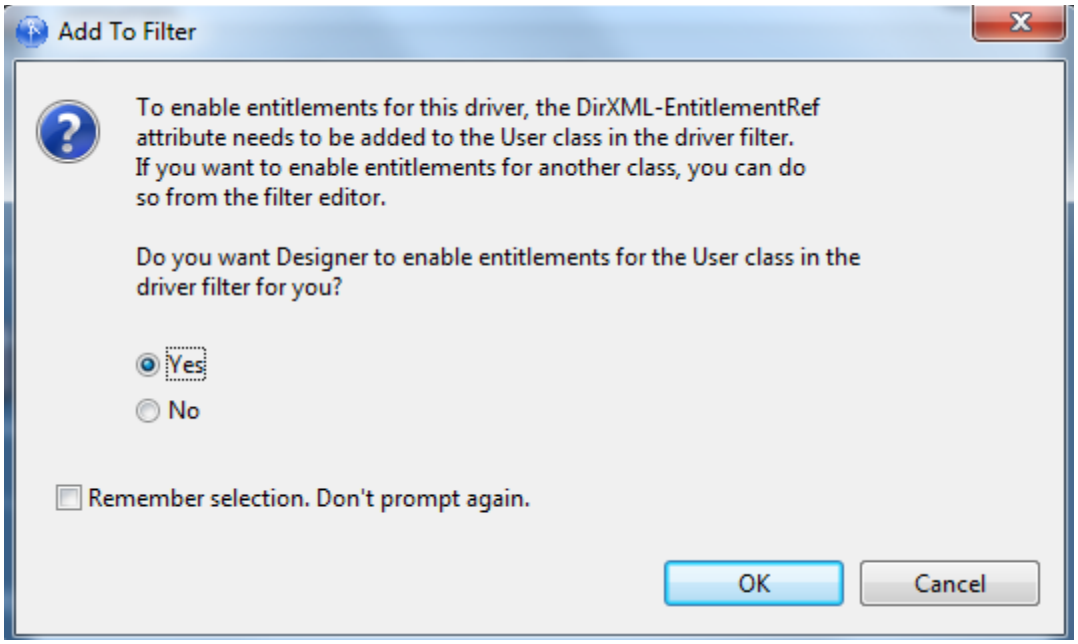
For this example, the query values look for the Source Distinguished Name attribute of the Class name of Group, starting from the Base DN (Blanston) and checking through the subtree from that beginning point. The values that come back from the query are similar to the following:

```
<instance class-name="Group" src-dn="o=Blanston,cn=group1">
   <association>o=Blanston,cn=group1</association>
   <attr attr-name="Description"> the description for group1</attr>
</instance>
<instance class-name="Group" src-dn="o=Blanston,cn=group2">
   <association>o=Blanston,cn=group2</association>
   <attr attr-name="Description"> the description for group2</attr>
</instance>
<instance class-name="Group" src-dn="o=Blanston,cn=group3">
   <association>o=Blanston, cn=group3</association>
   <attr attr-name="Description"> the description for group3</attr>
</instance>
<!-- ... ->
```

The information received from the query fills in the various fields. For instance, the <display-name> field receives o=Blanston,cn=group1. The <description> field receives the description for Group1, and the <ent-value> field receives o=Blanston,cn=group1. Because more than one group exists and meets the query criteria, this information is also collected and shown as other instances of the query.

The association format value is unique for every external system, so the format and syntax are different for each external system queried.

13 In the Add To Filter window, click **Yes** if you want the driver to listen for this entitlement. This enables entitlements for the driver.



The DirXML-EntitlementRef attribute allows the driver filter to listen for entitlement activities. This is necessary in order to use the entitlements you are creating.

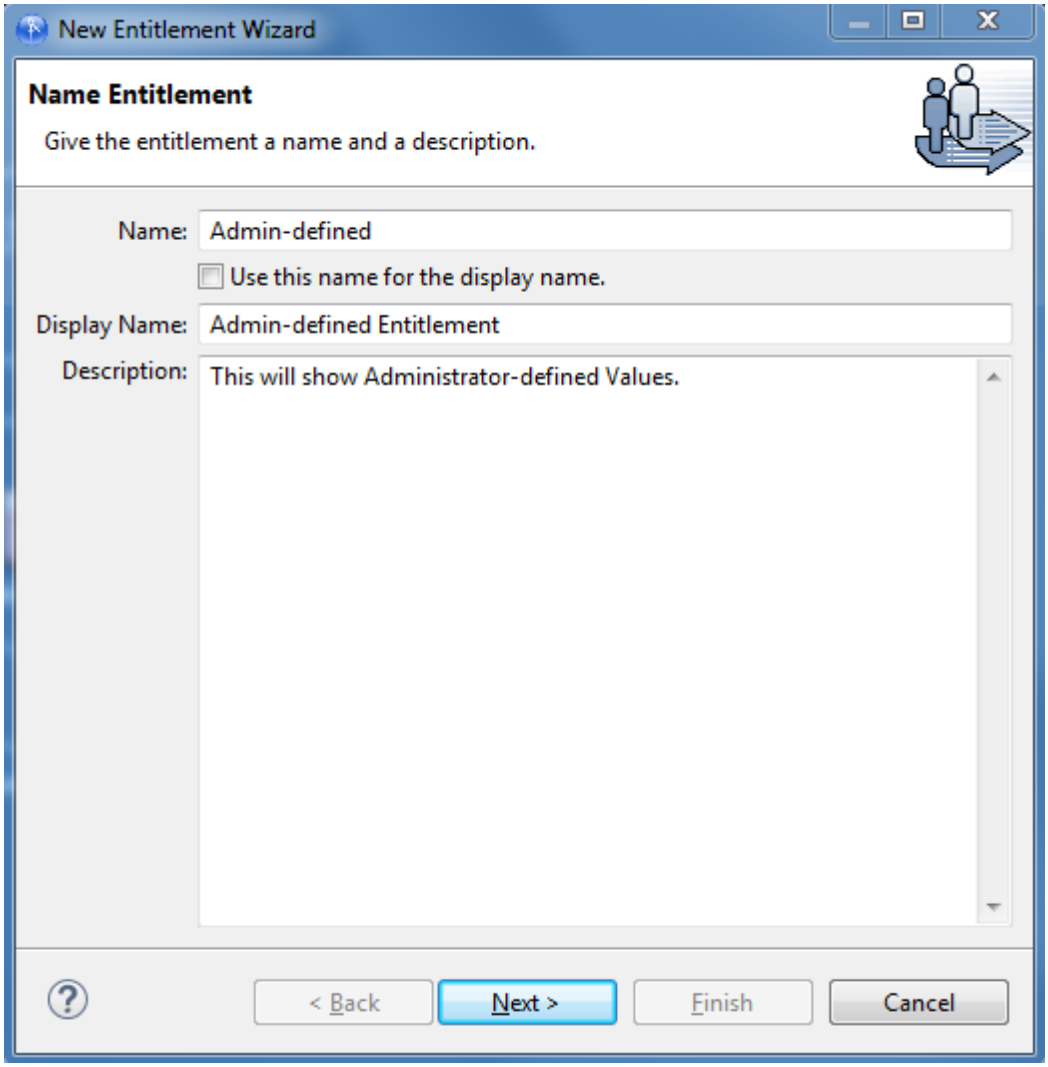
If you don’t want to see the Add To Filter window on entitlements you are creating for any driver in Designer, select **Remember Selection - Don’t Prompt Again**, then click **OK**. However, after the attribute is added to this driver filter, you won’t see the Add To Filter window again.

If you have a file conflict, you are asked to save the editor’s changes before continuing. When the editor is saved, the entitlement displays in the Modeler view.

Administrator-Defined Entitlements with Lists

The example in the following procedure is an administrator-defined entitlement that allows you to select a listed entry. This type of entitlement is best used through Workflow entitlements rather than Role-Based Entitlements.

1 Right-click the driver icon in the Outline view or the Modeler view, then click **New > Entitlement**.



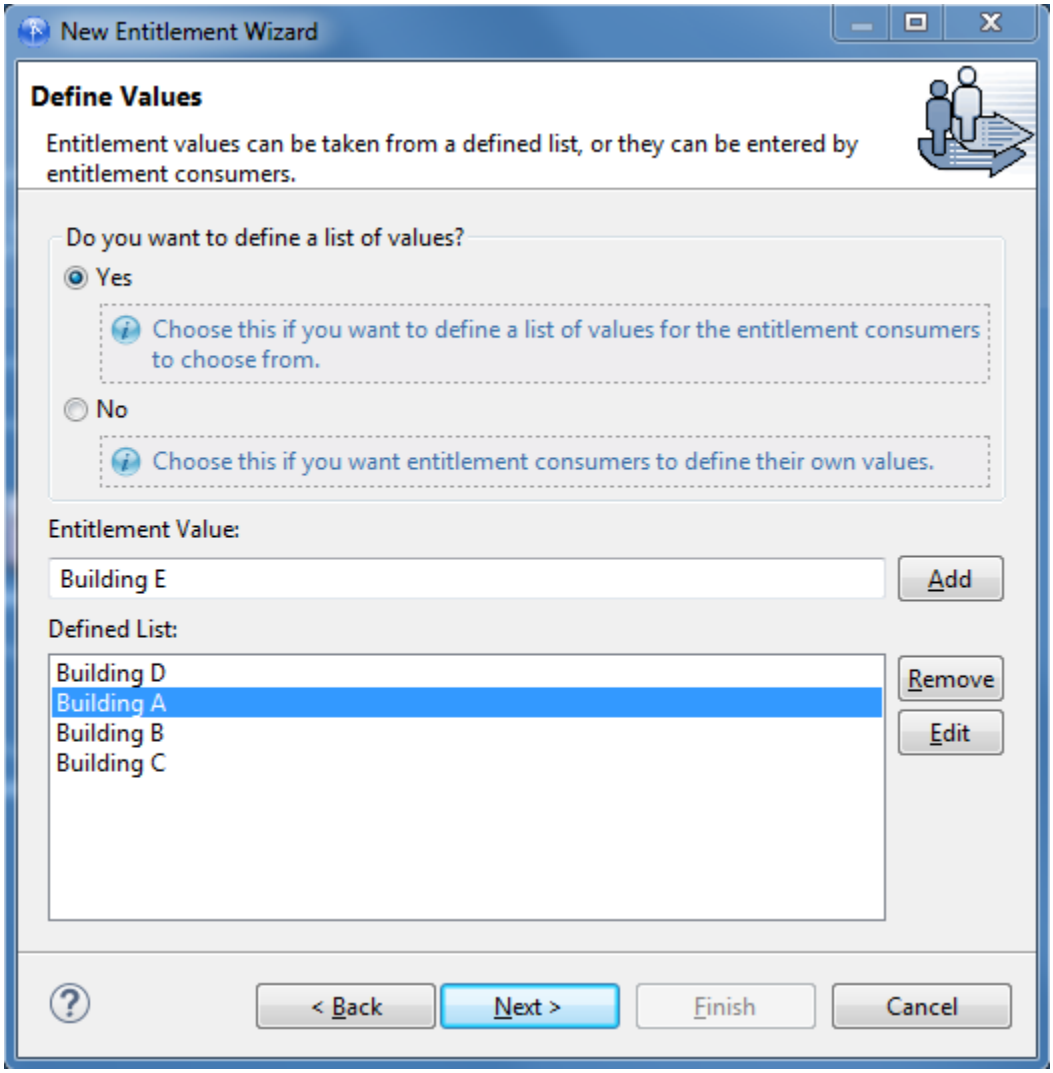
In this example, the entitlement name is Admin-defined, but the defined display name is Admin-defined Entitlement. You need to define a display name only if you want the display name to be different from the name you called the entitlement; otherwise, you can just use the entitlement name as the display name. In this example, the **Description** field is defined as **This will show Administrator-defined Values**.

2 Click **Next**.

3 In the Set Entitlement Values window, select **Yes** to the question “Do you want this entitlement to include values?” Click **Next**.

4 In the next Set Entitlement Values window, select **Administrator Defined Values**, then click **Next**.

5 In the Define Values window, type the values you want to add to the Entitlement Value entry, click **Add** to add the value to the Defined List pane, then click **Next**.



In this example, the values are corporate buildings: Building A through Building D. Through an entitlement client, such as an iManager Role-Based Entitlement task or through the user application, users or defined-task managers can specify the building information, which is then included in an external application, such as OpenText eDirectory.

Use the **Remove** icon to remove a value, or use the **Edit** icon to edit a value.

6 In the Assign Multiple Values window, select **Yes** if you want the entitlements to be granted more than once and with different values. If you select **No**, the entitlement can only be granted once. For the example, click **No**, then click **Next**.

It makes sense to assign group entitlements with multiple values, but it does not make sense to assign building letters more than once.

7 You are asked if this entitlement is intended to be used by Role-Based Entitlement policies through iManager. If you want this entitlement to be granted or revoked automatically, select **Yes** to the Role-Based Entitlements question, click **Next**, then continue with Step 8.

or

If you want the granting or revoking of this entitlement to be a manual process (approved by someone), select **No** to use the User Application, then skip to Step 9.

We recommend that you have only one agent control an entitlement. If multiple agents are in control, you have the following consequences:

- Whatever comes last controls the entitlement results
- Results are unpredictable
- Using both agents to control an entitlement is not supported by OpenText

8 (Conditional) If you select **Yes** to the Role-Based Entitlements question, you are asked if you want to use the Role-Based Entitlements priority to resolve any conflicts that might happen when this entitlement is assigned by different Role-Based Entitlement Policies with different values. You can resolve the conflict by either using the Role-Based Entitlements priority, or by merging the values. This example merges the values.

Merging the values merges the entitlements of all involved Role-Based Entitlement policies, so if one policy revokes an entitlement but another policy grants an entitlement, the entitlement is eventually granted. Solving conflicts by priority works if you need to ensure that only one policy is applied to this entitlement at any time.

9 Click **Finish**.

10 If you see the Add To Filter window, answer **Yes** if you want the driver to listen for this entitlement. This enables entitlements for the driver. The DirXML-EntitlementRef attribute allows the driver filter to listen for entitlement activities, which is necessary in order to use the entitlements you are creating.

or

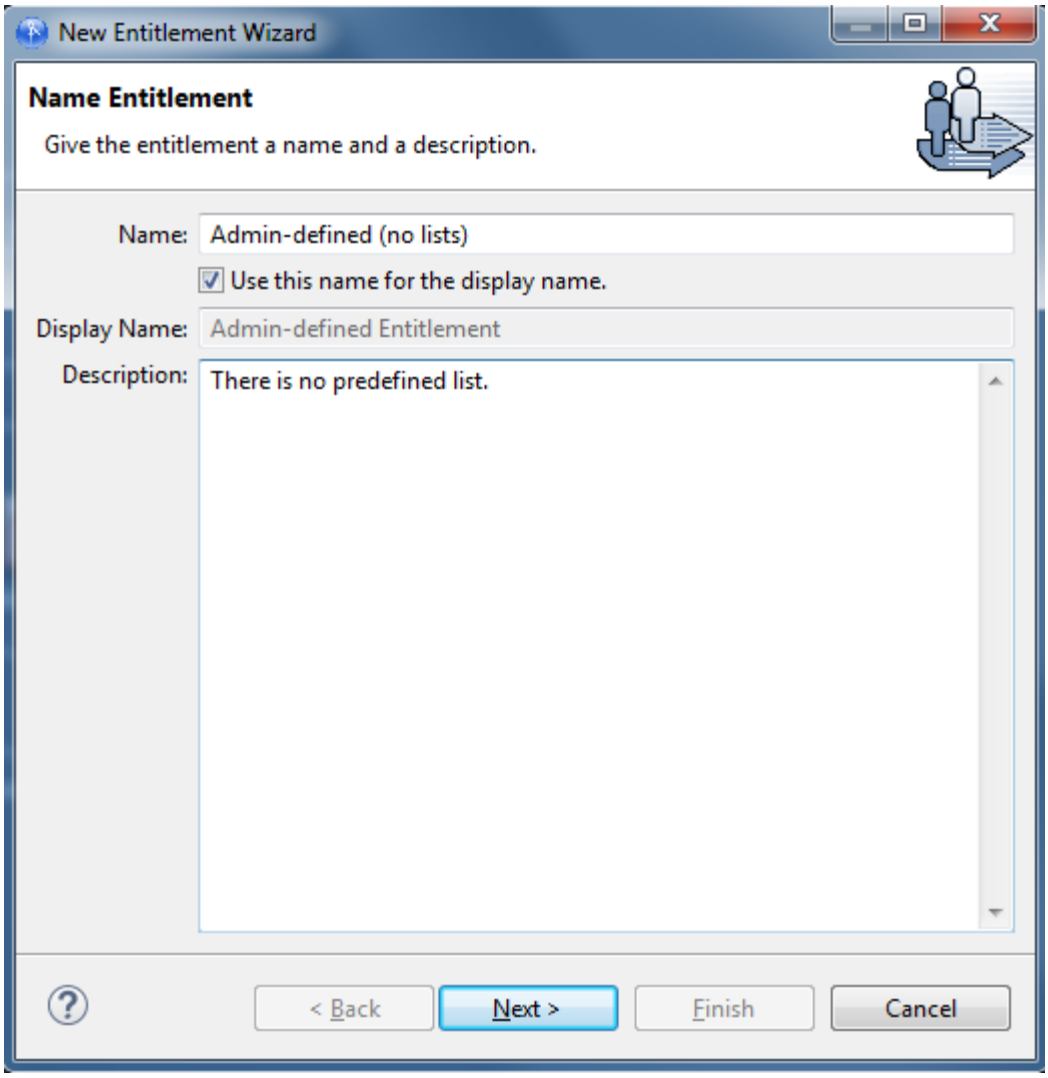
If you don’t want to see the Add To Filter window on entitlements you are creating for any driver in Designer, select **Remember Selection - Don’t Prompt Again**, then click **OK**. However, after the attribute is added to this driver filter, you won’t see the Add To Filter window again.

Before you can edit this entitlement, you are asked to save the editor’s changes before continuing. When the editor is saved, the entitlement displays in the Modeler view.

Administrator-Defined Entitlements without Lists

The example in the following procedure is an administrator-defined entitlement that forces the administrator to type a value. You can use this kind of entitlement if you cannot create a task list because you do not have all of the information at the initial setup.

1 Right-click the driver icon in the Outline view or the Modeler view, then click **New > Entitlement**.



In this example, the entitlement name is **Admin-defined (no lists)**, and it uses the entitlement name as the displayed name because the **Use Name For Display Name** option is selected.

2 Click **Next**.

3 Select **Yes** on the Set Entitlement Values page, then click **Next**.

4 Select **Administrator Defined Values** on the second Set Entitlement Values page, then click **Next**.

5 Select **No** to the question "Do you want to define a list of values?" on the Define Values page, then click **Next**.

Selecting this option allows the administrator or users to type a value.

Be aware that using this option can be risky, because wrong or misspelled information can cause the value to be incorrect and the action in the entitlement to fail.

6 Select **No** to the question "Allow this entitlement to be assigned multiple times with different values?" on the Assign Multiple Values page, then click **Next**.

7 You are asked if this entitlement is intended to be used by Role-Based Entitlement policies through iManager. If you want this entitlement to be granted or revoked automatically, select **Yes** to the Role-Based Entitlements question, click **Next**, then continue with Step 8.

or

If you want the granting or revoking of this entitlement to be a manual process (approved by someone), select **No** to use the User Application, then skip to Step 9.

We recommend that you have only one agent control an entitlement. If multiple agents are in control, you have the following consequences:

- Whatever comes last controls the entitlement results
- Results are unpredictable
- Using both agents to control an entitlement is not supported by OpenText

8 (Conditional) If you select **Yes** to the Role-Based Entitlements question, you are asked if you want to use the Role-Based Entitlements priority to resolve any conflicts that might happen when this entitlement is assigned by different Role-Based Entitlement Policies with different values. You can resolve the conflict by either using the Role-Based Entitlements priority, or by merging the values. This example uses priority.

Merging the values merges the entitlements of all involved Role-Based Entitlement policies, so if one policy revokes an entitlement but another policy grants an entitlement, the entitlement is eventually granted. Solving conflicts by priority works if you need to ensure that only one policy is applied to this entitlement at any time.

9 Click **Finish**.

10 If you see the Add To Filter window, answer **Yes** if you want the driver to listen for this entitlement. This enables entitlements for the driver. The DirXML-EntitlementRef attribute allows the driver filter to listen for entitlement activities, which is necessary in order to use the entitlements you are creating.

or

If you don't want to see the Add To Filter window on entitlements you are creating for any driver in Designer, select **Remember Selection - Don't Prompt Again**, then click **OK**. However, after the attribute is added to this driver filter, you won't see the Add To Filter window again.

Before you can edit this entitlement, you are asked to save the editor's changes before continuing. When the editor is saved, the entitlement displays in the Modeler view.

Editing and Viewing Entitlements

After you have created entitlements, you might need to edit them. You can also use the Edit mode to see the entitlements in their XML source code.

- "Entitlement XML Source and XML Tree Views" on page 369
- "Using the Entitlement DTD" on page 372

To edit an entitlement:

1 From the Outline view, right-click an entitlement that appears under the Subscriber channel of the selected driver, then click **Edit**.

or

Double-click the entitlement icon to bring up the entitlement in the Entitlement editor.

You can also right-click the driver icon in the Modeler view, then select **Edit Entitlements**.

2 If you have more than one entitlement for the selected driver, you see the Edit Entitlements windows listing the available entitlements. Select an entitlement, then click **OK**.

The entitlement appears in the Entitlement editor.

The Entitlement Editor view shows you all of the pages and choices that you see in the Entitlement Wizard, but the information is on one page.

- **Entitlement Editor:** Displays the full DN name for the entitlement. If there is a conflict with the entitlement name or some other error, you see a red icon to the left of the Entitlement editor name, followed by an error message.
- **Name and Description:** Allows you to edit the name, the display name, and the description that you have given to this entitlement.
- **Multi-Value:** Allows you to select if you want an entitlement to be assigned multiple times.
- **Role-Based Entitlements:** Allows you to select conflict resolution for Role-Based Entitlements. If you do not select Role-Based Entitlements, **the Role-based entitlements with priority** icon is the default.
- **Values:** Allows you to define how values are defined: no values, administrator defined values, or values from an application.

The information that appears in the Entitlement editor depends on what you initially defined in the entitlement. If you choose to edit a valueless entitlement, the **Values** heading displays **No Values**. If you are editing a valued entitlement and you want to add values to a list, type the value in the **Value** field and click **Add**. If you want to remove a value, select the value in the **Values** list and click **Remove**.

If you don't want to select from a list, select **Administrator Defined Values** under the **Values** heading and leave the **Values** list blank. This gives you a blank text box in iManager or in the user application, and you can fill in the value there.

3 When you have made your changes to the entitlement, click the **Save** icon in the upper left corner of Designer, or click the **X** on the entitlement's tab to display a Save Resource window, allowing you to save changes (**Yes/No/Cancel**).

Entitlement XML Source and XML Tree Views

To view the entitlement in XML source code:

1 From the Outline view, right-click an entitlement that appears under the Subscriber channel of the selected driver, then click **Edit**.

or

Double-click the entitlement icon to bring up the entitlement in the Entitlement editor.

You can also right-click the driver icon in the Modeler view, then select **Edit Entitlements**.

2 To see the XML Source view, click **XML Source** at the bottom of the Entitlement Editor view.

The XML Source view shows the XML code in a formatted state.

The upper right corner of the XML Source view has the following selections:

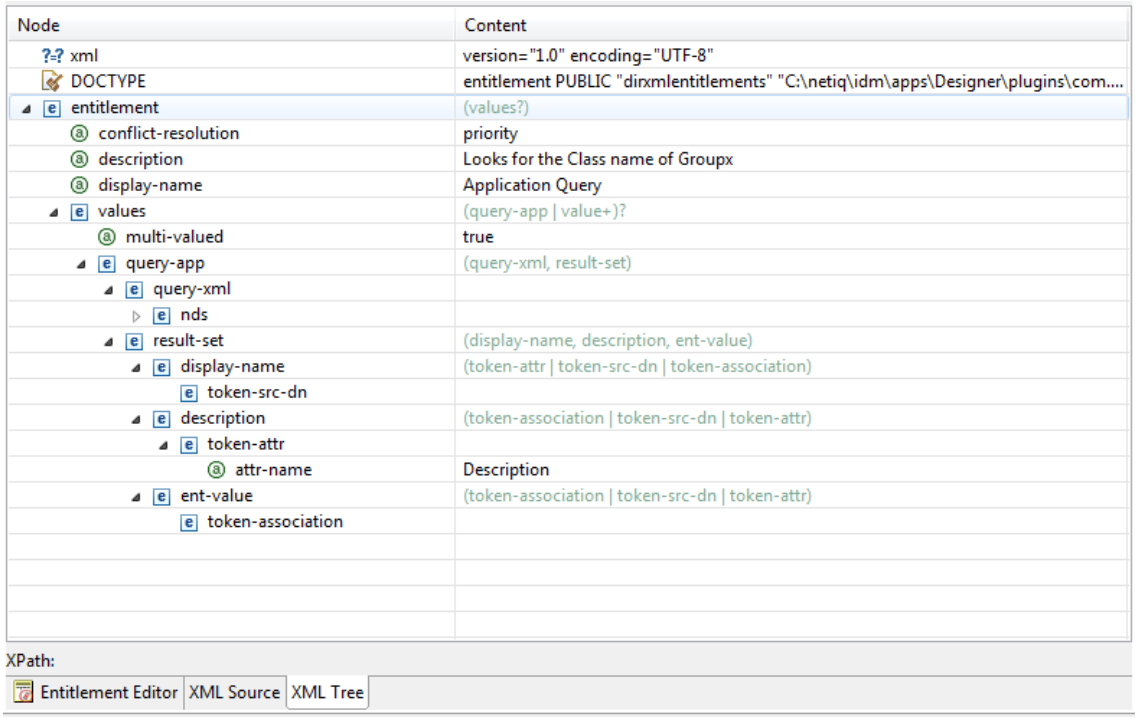
Name	Description
Expand All	Allows you to see all items under the item that you have selected.
Collapse All	Allows you to collapse all items that you have selected.
Attach XML Catalog Entry, XML Schema, or DTD	Allows you to attach an XML Catalog entry, an XML schema file, or a DTD (Document Type Definition) file. For default Windows installation, the DTD for entitlements is found under `C:\Program Files\Novell\Designer\eclipse\plugins\com.novell.designer.idm.entitlements_1.1.0\DTD\dirxmlentitlements.dtd`.
Copy XML to Clipboard	Allows you to copy highlighted XML code to the clipboard. This action removes the DOCTYPE element.
Find/Replace (Ctrl+F)	Ctrl+F brings up the Find/Replace window, which allows you to query text, structure, and XPath searches in a forward or a backward direction. Other options include case sensitive, wrap search, whole word, incremental, and regular expressions search capabilities.
Help	Opens the Help view to the right of the XML Source view.

Right-clicking in the XML Source view brings up the following options:

- Undo Text Change (Ctrl+Z)
- Revert File
- Save
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Format the document or active elements
- Clear Validation Errors
- Validate
- Preferences

3 To see the XML Tree view, click **XML Tree** at the bottom of the Entitlement Editor view.

The XML Tree view is a tree control view of the XML source code. You can perform the same edits in this view as you can in the Entitlement Editor view or the XML Source view. To view the entitlement in XML Tree view, select **XML Tree** at the bottom of the Entitlement Editor view.



The upper right corner of the XML Tree view menu contains the following selections:

Name	Description
Expand All	Allows you to see all items under the item that you have selected.
Collapse All	Allows you to collapse all items that you have selected.
Attach XML Catalog Entry, XML Schema, or DTD	Allows you to attach an XML Catalog entry, an XML schema file, or a DTD (Document Type Definition) file. For default Windows installation, the DTD for entitlements is found under `C:\Program Files\Novell\Designer\eclipse\plugins\com.novell.designer.idm.entitlements_1.1.0\DTD\dirxmlentitlements.dtd`.
Find/Replace (Ctrl+F)	Brings up the Find/Replace window, which allows you to query text, structure, and XPath searches in a forward or a backward direction. Other options include case sensitive, wrap search, whole word, incremental, and regular expressions search capabilities.
Help	Opens the Help view to the right of the XML Tree view.

Right-clicking in the XML Tree view can bring up a number of different options. For example, right-clicking the highlighted value on the right side presents the following options:

- Undo
- Cut
- Copy
- Paste

- Delete
- Select All

Right-clicking an attribute on the left side in the XML Tree view presents the following options:

- Remove
- Edit the Selected Attribute
- Replace with a value

Depending on what you select on the left side in the XML Tree view, you see different options. For example, right-clicking an element presents the following options:

- Remove Element
- Add New Attribute
- Add to a Child Element a Comment, a Processing Instruction, a PCDATA, a CDATA Section, a new Element
- Add Before a Comment, a Processing Instruction, a PCDATA, a CDATA Section, a new Element
- Add After a Comment, a Processing Instruction, a PCDATA, a CDATA Section, a new Element

Using the Entitlement DTD

Some entitlements come predefined on drivers that have entitlements enabled. (For a list of these drivers with predefined entitlements see "OpenText Identity Manager Drivers with Preconfigurations that Support Entitlements" on page 352.) You can use these entitlements or you can create your own entitlements in iManager or Designer. To help you create your own entitlements, you can use the Entitlement DTD as an example to create entitlements. For an example of the Entitlement DTD and an explanation of its functionality, see the "Writing Entitlements in XML" section of the *OpenText™ Identity Manager Entitlements Guide*.

Managing Entitlements

After you create entitlements (or use entitlements that come preconfigured with certain OpenText Identity Manager drivers), you need to manage them. Entitlements are tied into the OpenText eDirectory event system and granting and revoking are initiated through two agents:

- iManager through Role-Based entitlement policies
- The User Application as workflow entitlements

Role-Based Entitlements allow you to automatically grant or revoke business resources if the criteria are met. In order for workflow entitlements to work with the User Application, manual approval is first required.

For instance, you can specify that if user has A, B, and C qualification, then the user is made a member of Group H; but if the user has E and F qualifications, he or she is made a member of Group I. Through Role-Based Entitlements, this action is done automatically, as long as the conditions are met. In order for this entitlement to work with workflow entitlements, the User object must first acquire approval, which you need to set up through the User Application. However, if you do not add to the driver the policies and rules to interpret the event in the designated system, granting and revoking entitlements has no effect.

Use either Role-Based Entitlements or workflow entitlements. It is a not good idea to mix them to manage the same resource. We recommend that you have only one agent control an entitlement. If multiple agents are in control, you have the following consequences:

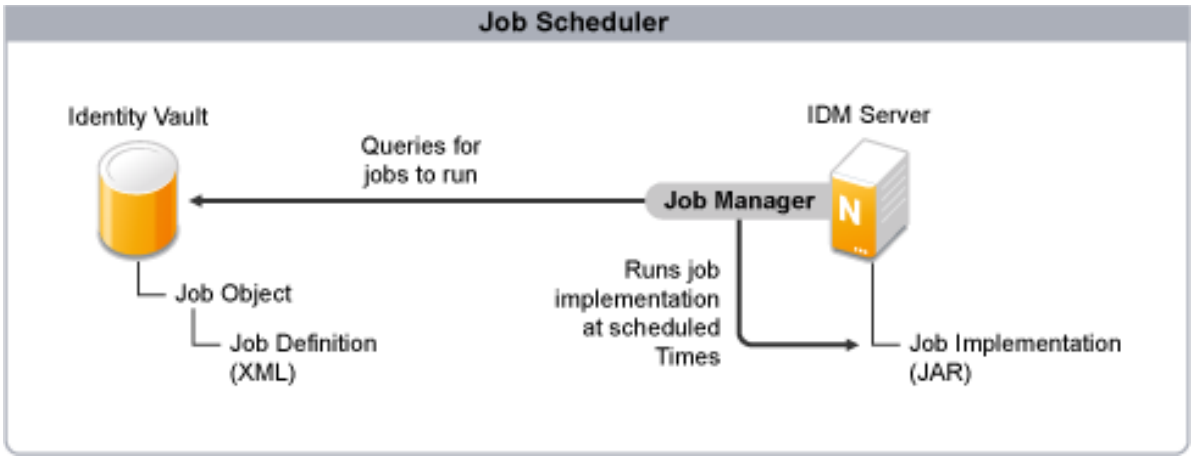
- Whatever comes last controls the entitlement results
- Results are unpredictable
- Using both agents to control an entitlement is not supported by OpenText

14 Scheduling Jobs

Designer has a job scheduling utility to schedule events, such as setting the system to disable an account on a specific day, or initiating a workflow to request an extension for a person's access to a corporate resource. You can use it to do the following tasks:

- Create a Job object from an installed job definition.
- Define when a job is to run, which servers the job is to run on, the scope of the job in terms of OpenText eDirectory objects, and the job reports for intermediate and final results.
- Set values for the job's parameters, its description, and display name.
- Enable or disable a job, manually start a job, stop a job that is running, and display a list of running jobs.

Figure 14-1 *High-level View of the Job Scheduler Process*



- "Job Scheduler Components" on page 375
- "Creating a Job" on page 376
- "Editing a Job" on page 377

Job Scheduler Components

The Job Scheduler consists of the following principal components:

Job Manager: Responsible for launching scheduled jobs. It runs in the background on each OpenText Identity Manager server and checks every minute to see if a job needs to run, based on the job definition. When it encounters a job that needs to run, the Job Manager runs the appropriate Job Implementation.

Job Object: An object you create in Designer. It contains all the information necessary to invoke the job, including the name, description, schedule, server list, and XML job definition.

Job Definition: An XML description of all the parameters necessary to perform a specific job, including the Job Implementation used to actually perform the job on the target servers. The Job Definition is an XML attribute associated with the Job Object.

Job Implementation: A JAR file that contains the Java classes that perform the job on the target OpenText Identity Manager server. Each server where you want a job to run must have a copy of the Job Implementation file. At the designated time, as specified in the Job Definition, Job Manager runs the Job Implementation to perform the job.

Creating a Job

1 In the Outline view, right-click a driver and select **New > Job**.

You can also right-click a driver set and select **New > Job** to create a driver health job. For more information about driver health configuration and the driver health job, see "Driver Health Configuration" on page 93.

This opens the New Job page.

2 In the **Names** field, specify a descriptive name for the job, or use the default name provided.

3 Select **Installed** to create a job using an existing job definition, or select **Custom** to create a custom job definition for this job.

3a If you are creating a job from an existing job definition, select the job definition you want to use from the list of available jobs.

The New Job Wizard comes with three job definitions.

- **Random Password Generator:** Generates a random password for each object in the job's scope. The password is generated by NMAS to match the Password Policy object that the job references. If policy-dn is not specified, the effective password policy of the current object in OpenText eDirectory is used. If the current object does not exist (for example, the target of an add operation on the publisher channel), the effective password policy of the target container is used. These Password Policy objects are not usually the same as those used for OpenText eDirectory user password policies.

  The job submits the generated passwords one at a time to the driver's Subscriber channel. The Subscriber channel policies must do something useful with the passwords.

- **Schedule Driver:** Starts or stops the associated driver. You can also toggle a driver to start the driver if it is stopped or to stop the driver if it is running.

- **Subscriber Channel Trigger:** Submits zero or more trigger documents to the Subscriber channel. The submission can either be a document per object if a scope is defined, or it can be a single trigger event if no scope is defined.

  Trigger event documents identify the job and the scope object. A trigger event can bypass the cache and go to the head of the queue if desired. You will probably use trigger jobs the most; they allow you to use driver policies that you can customize for your personal requirements.

Click the **Update Job Definitions from Server** icon  to display any custom job definitions on the selected server. Because Designer is an offline modeling tool, only the OpenText Identity Manager job definitions display by default.

3b If you are creating a custom job definition, paste the job definition XML into the code field.

The code field is not designed for entering XML directly, although you can do so if desired. OpenText Identity Manager provides a Job Scheduler DTD that defines the XML structure for job definitions. For more information, see Jobs DTD in the *OpenText Identity Manager DTD Reference*

The Job Scheduler automatically validates the custom job XML against the DTD specified in the content, or against the default Job Scheduler DTD if none is specified. It marks any errors it finds so you can review them, and requires you to fix serious errors before allowing you to save the custom job.

4. In the **Run Jobs on Servers** field, select the servers where you want to run the job.
5. Select **Edit Job configuration after creating the object** if you want Designer to open the newly created job in the Job Editor window after saving the job object.
6. Click **OK**.

   The File Conflict window informs you that you must save the job object to continue.
7. Click **Yes** to save the job and continue.
8. Continue with "Job Editor Selections on the General Settings Page" on page 378.

Copying a Job

There are two ways to create a new job based on an existing job:

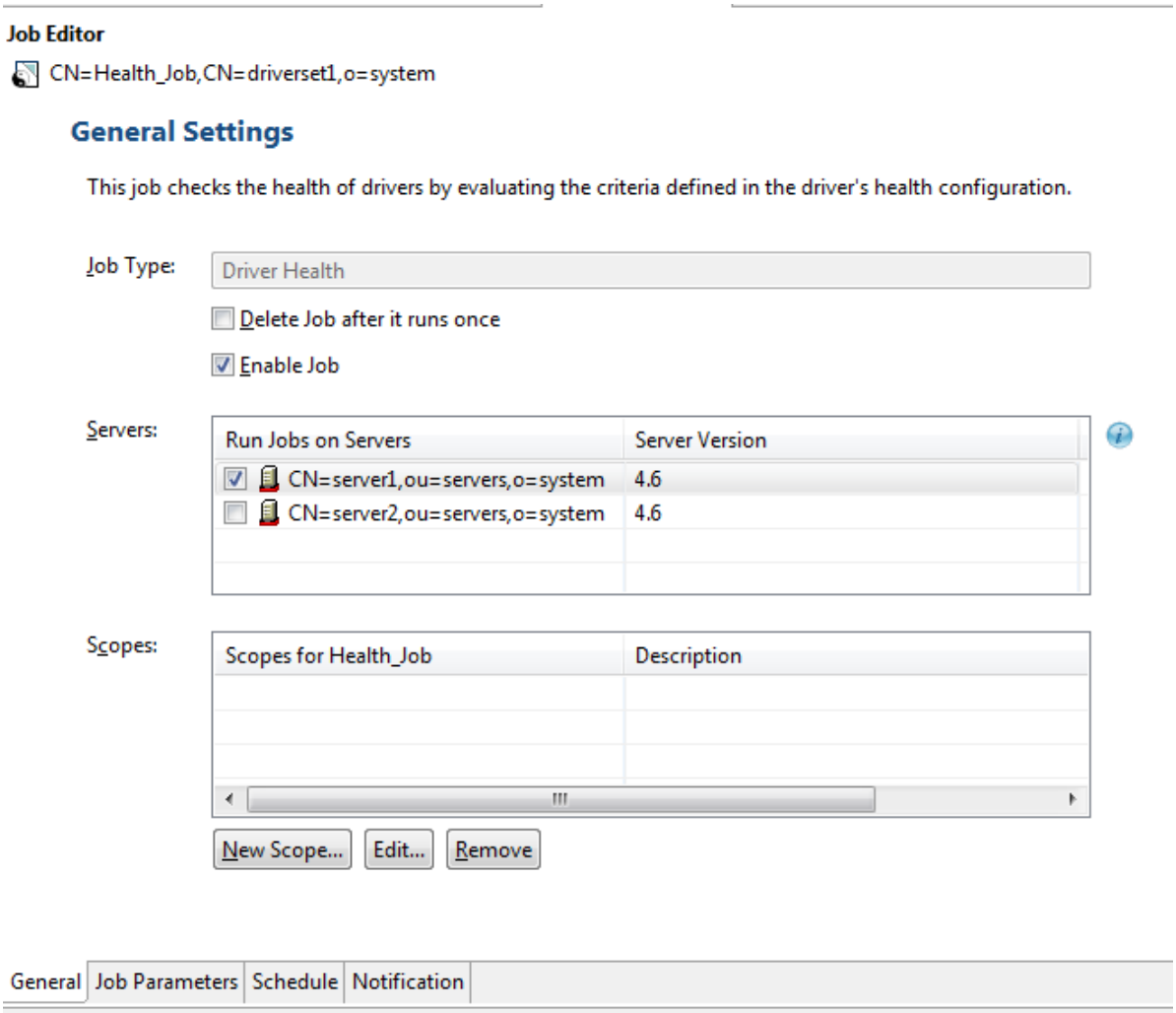
- In the Outline view, right-click an existing job object, then select **Copy**. This creates a duplicate job object in the same location as the original job object.
- Right-click a driver, then select **New** > **Copy From**. This is useful if you want to create a job in a different location from the original job object., such as in a different driver.

In either case, after you create the new job object, you can then edit the job as needed to fit your needs. For more information, see "Editing a Job" on page 377.

Editing a Job

After you create a job, you need to add the necessary information to make the job useful. To edit a job, double-click a newly created job in the Outline view to bring up the job in the Job Editor view.

Figure 14-2 *The Job Editor View*



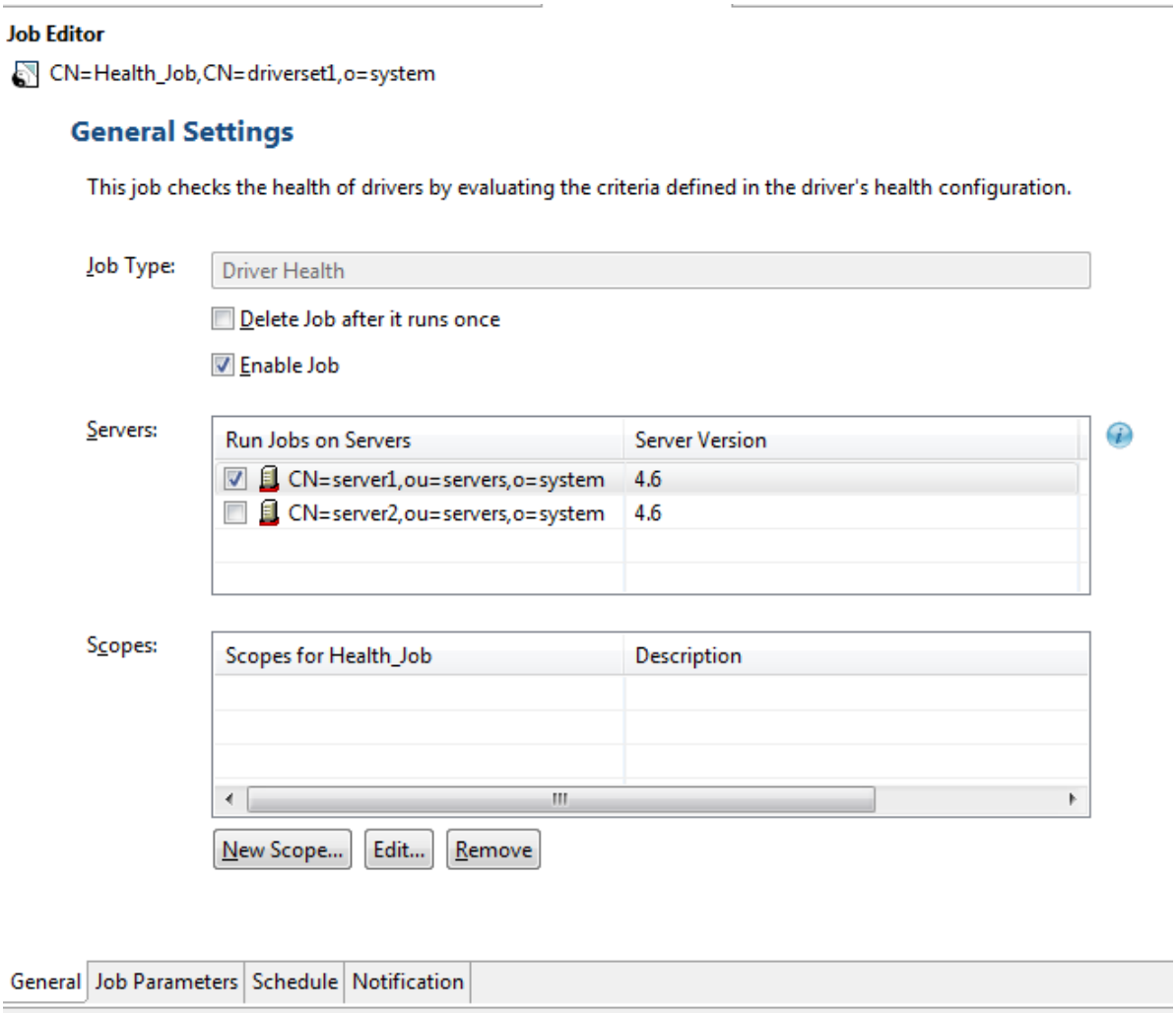
The Job Editor has four tabs at the bottom of its view:

- “Job Editor Selections on the General Settings Page” on page 378
- “Job Editor Selections on the Job Parameters Page” on page 381
- “Job Editor Selections on the Scheduler Page” on page 381
- “Job Editor Selections on the Notification Settings Page” on page 383

Job Editor Selections on the General Settings Page

The title of the General Settings Page shows the Java class name of the job. This is followed by the job type, which shows the type of job you selected. Under the **Job Type** heading, you can enable or disable the job, or delete the job after it runs.

Figure 14-3 *General Settings Page*



1 To delete the job after it runs, select **Delete Job after it runs once**.

2 To disable the job from running, deselect **Enable Job**.

3 In the **Servers** column, select the server or servers where this job should run.

   A filtered list of servers is available to help you assign this job. A custom job can be installed on one server but not on another. In this case, the server without this custom job is filtered out of the Server List.

   A job can be assigned to multiple servers as long as it has been installed on each server. Designer only allows this association if the jobs are properly installed and packaged so that the engine can see them.

4 To add a scope to the **Scopes** column, click **New Scope**.

5 To select a scope object, type the Distinguished name of the object or use the **Browse** icon to browse to the object. Click **OK** to add the scope object.

Scopes allow you to define the objects that this job applies to. An object in OpenText eDirectory can be a container, a dynamic group, a group, or a leaf object. If you select a group object, you can apply the job to the group's members, or only to the group. If you select a container object, you can apply the job to all descendants in that container, to all of the children in the container, or to the container only.

6 If the object is a container, select **Scope is a Container**. Then select how you want to apply the job:

- Apply job to this container only
- Apply job to children of this container
- Apply job to all descendants of this container

7 (Optional) If you select **Apply job to children of this container** or **Apply job to all descendants of this container**, you can specify the classes and attributes you want to scope. Click the plus icon to bring up the Schema Browser window to select the classes you want to scope. Select the class schema, then click **OK**.

The classes are added to the Classes box. To remove a class, select it and click the minus icon.

8 If the object is a group or a dynamic group, select **Scope is a Group/Dynamic Group**. You can then select the **Scope is the group itself and not its members** option if the scope is for the group.

9 If the object is a non-container, select **Scope is a Non-Container**.

10 After the scope criteria are selected, click **OK** to return to the General Settings page.

11 If you need to edit a scope, select the scope name, then click **Edit**.

12 To remove a scope, select the scope name, then click **Remove**.

Deploying a Job with Scope Objects

Jobs might need access to OpenText eDirectory data and certain OpenText Identity Manager actions, such as starting and stopping drivers. Such access is subject to OpenText eDirectory rights assignments and is controlled by the rights that are granted to the DirXMLJob object. Although OpenText Identity Manager actions are controlled by special attributes, normal OpenText eDirectory rights are needed for data reads and writes.

When you deploy a job object that has scope objects, there might be OpenText eDirectory rights assignments that Designer cannot properly set up. The rights needed to complete the task depend on the scope objects that are assigned to the job object.

If you see this warning when deploying job objects, use the iManager utility to assign OpenText eDirectory rights to the job object so it can properly access the job scope objects and complete its task.

When you deploy a job for the first time, the Deploy - Security Equivalences dialog is displayed. To add or delete security equivalences, click **Add** or **Delete**.

Job Editor Selections on the Job Parameters Page

The Job Parameters page allows you to add additional parameters to the job and to view the parameters as they are presently set up. What you can do depends on the type of job you selected.

- "Parameters for the Generate Random Passwords Job" on page 381
- "Parameters for the Subscriber Channel Trigger Job" on page 381

NOTE: The parameters for a custom job vary based on the job's design. For more information about creating a custom job, see "Creating a Job" on page 376.

---

1. If you want the job to start the driver, select **Start the driver**.
2. If you want the job to stop the driver, select **Stop the driver**.
3. If you want the job to switch from one to the other, select **Toggle the driver**.

Parameters for the Generate Random Passwords Job

1. Type the Password policy object's Distinguished name, or use the **Browse** icon to select the Password policy you want to use for password generation.
2. If you want to generate passwords for scoped objects without a driver association, select **True**. Otherwise, select **False**.

Parameters for the Subscriber Channel Trigger Job

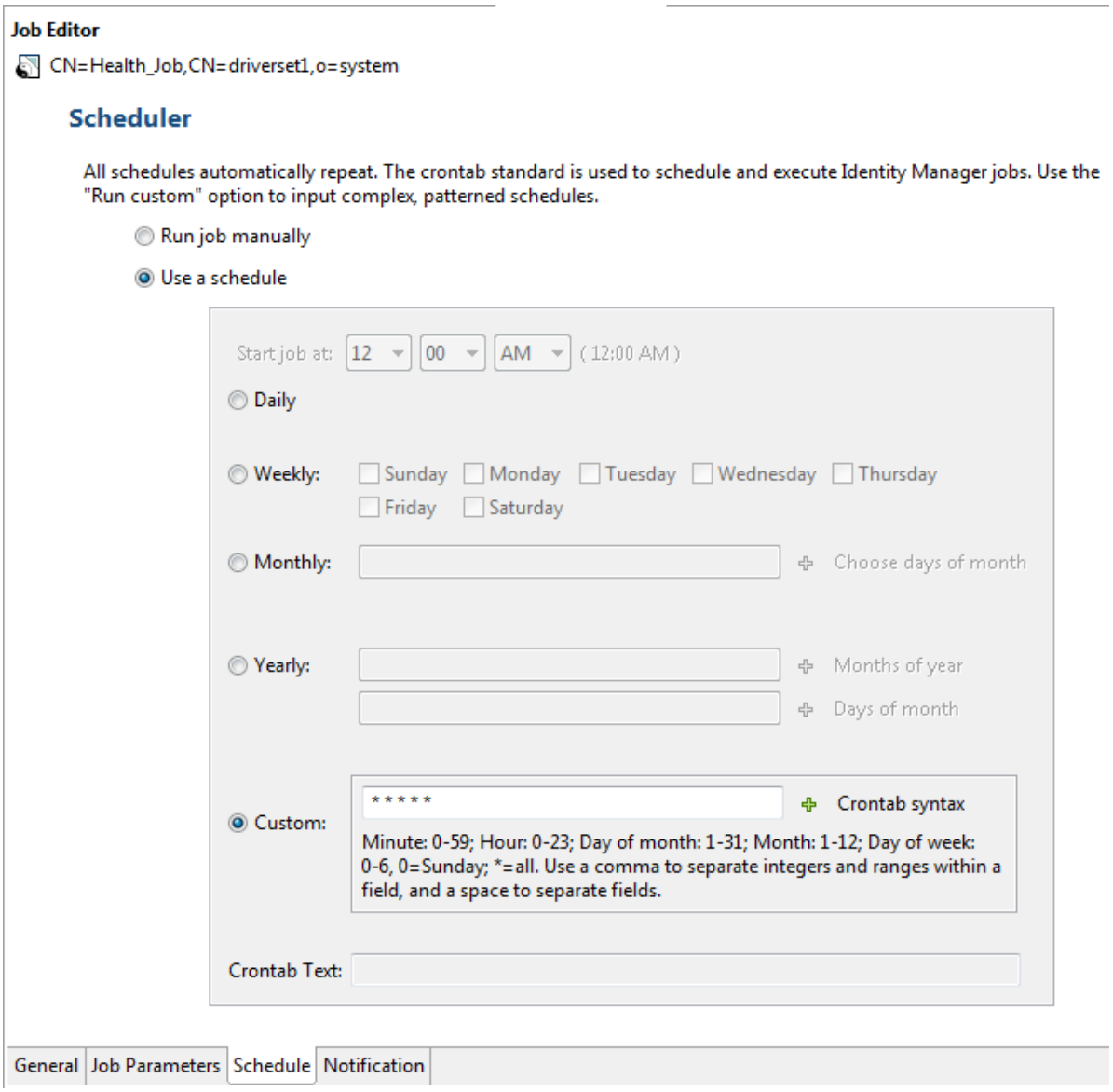
1. If you want to submit a trigger document for scoped objects that do not have a driver association, select **True**. Otherwise, keep the default of **False**.
2. If you want to use the job's Common Name (CN) as a document identifier trigger, keep the default of **True**. Otherwise, select **False**.
3. (Optional) If you select **False**, specify the string that the job can use as the value for the trigger element's Source attribute.
4. Select a method for submitting the trigger documents. If you want to queue the job the trigger is from, keep the default of **Queue (use cache)**. Otherwise, select **Direct (bypass cache)**.
5. (Optional) If you select **Direct (bypass cache)**, you are presented with the **Start driver if not running** option. If you want to start the driver if it is not running, keep the default of **True**. Otherwise, select **False**.
6. (Optional) If you select **True** on the **Start driver if not running** option, you are presented with the **Stop driver when finished processing triggers** option with the default of **True**. Use the default to stop the driver when it finishes processing the trigger job, or select **False** to keep the driver running.

A customized job definition has its own parameter set.

Job Editor Selections on the Scheduler Page

The Scheduler page allows you to set up when you want to run the job.

Figure 14-4 *The Job Options for the Scheduler Page*



1 Select the **Use a schedule** option to set the date and time, and whether to run the job daily, weekly, monthly, yearly.

   or

   Select the **Run job manually** option to run the job when you choose to.

2 With **Use schedule** selected, set the time when you want the job to start running. Use the drop-down menus to select the hours, minutes, and AM or PM. The default is 1:00 AM.

3 If you want to run the job repeatedly, use the **Daily**, **Weekly**, **Monthly**, **Yearly**, or **Custom** fields to select when you want it to run.

   For example, if you want the job to run weekly, select **Weekly**, then the day you want it to run on. If you want the job to run once a month, select **Monthly**, then click the plus icon to select the day of the month.

4 (Optional) Select **Custom** to choose minutes, hours, days, months and days of the week from the Choose Advanced Crontab Criteria page.

5 The Choose Advanced Crontab Criteria page default has everything selected. Click **Unselect All**, choose the time and days you want to run the job, then click **OK** to return to the Scheduler page.

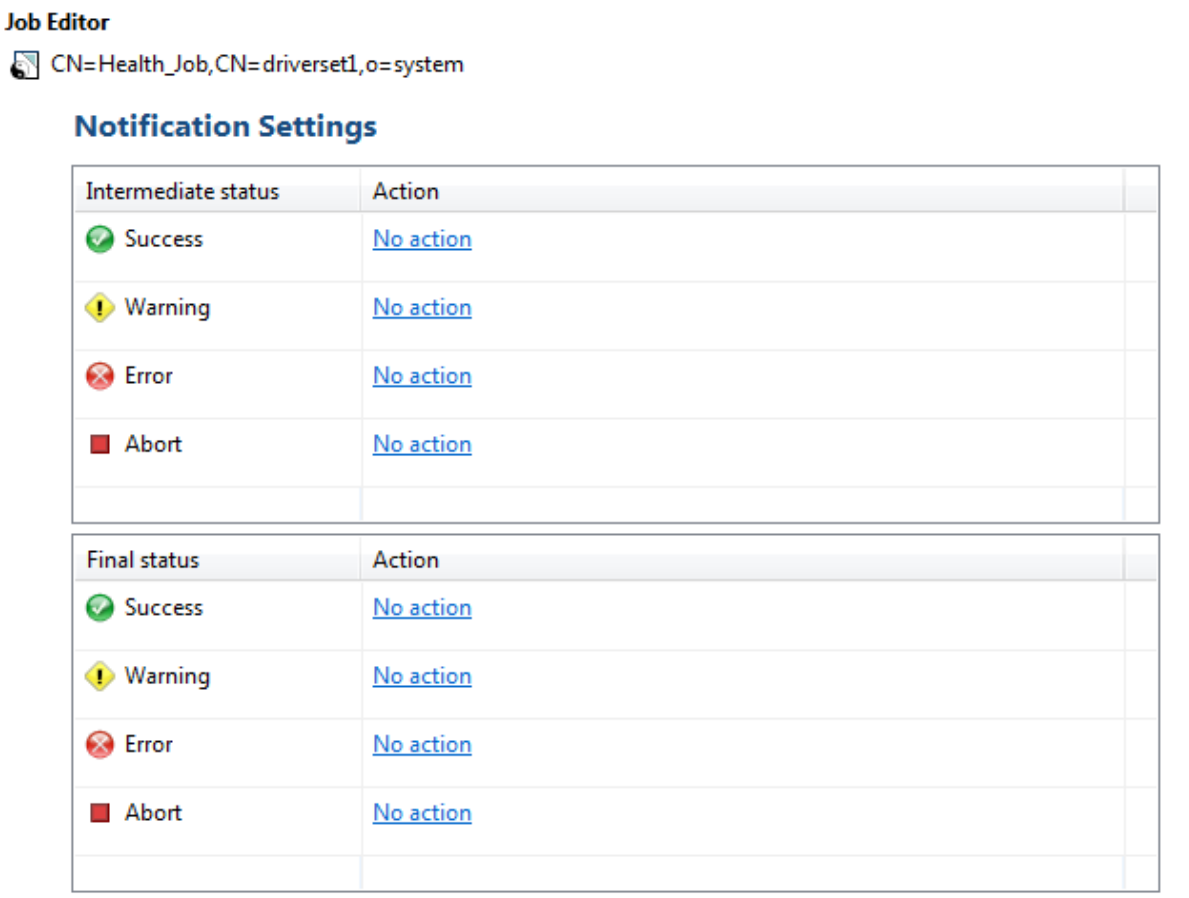
The information displayed in the **Crontab Text** field displays any settings you make on the Scheduler page. For example, if you click **Monthly** and select two days, those two days are displayed in the **Crontab Text** field.

Job Editor Selections on the Notification Settings Page

The Notification Settings page allows you to define what you want to do with the job results. It is divided into two parts, Intermediate and Final, with the Success, Warning, Error, and Aborted results for each part.

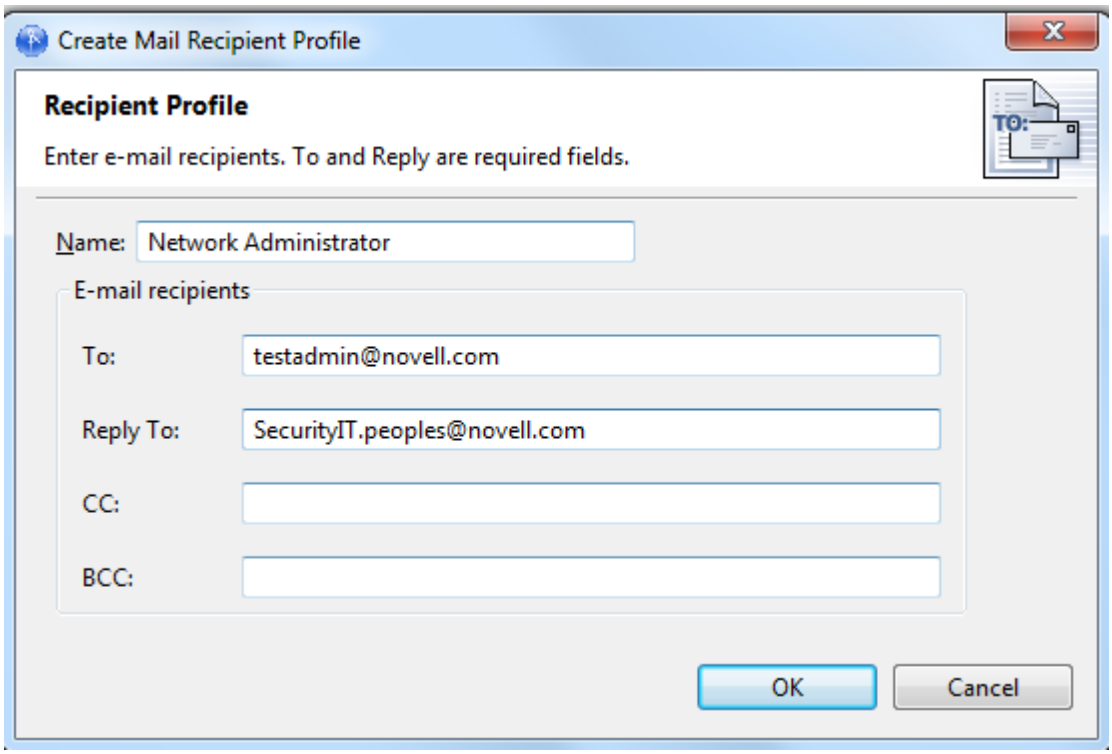
The Notification Settings page allows you to set how you want to be notified for each result. Actions include sending an audit result or sending an e-mail when the result completes.

Figure 14-5 *Notification Settings Page*



1 If you select **Send email for this event**, Designer allows you to search in the **Default Notification Collection** directory for an appropriate template to use in the **Notification Template** field. Click the **Model Browser** icon to select an appropriate template.

2 Under Notification Recipients, select who you want to send the results to by typing the user's or group's fully distinguished name. You can use the plus icon to create a mail profile or click the **Model Browser** icon to choose a mail profile.



The To and Reply fields are required for a profile.

3 When you have filled in the information, click **OK**.

4 If you want the results to go to Audit, select **Use NetIQ Audit for this event**.

5 Use Step 1 through Step 4 for each of the options:

- Intermediate Success
- Intermediate Warning
- Intermediate Error
- Intermediate Abort
- Final Success
- Final Warning
- Final Error
- Final Abort

If you do not select an option, no action is taken for the result.

15 Deploying and Exporting

The Deploy feature in Designer places a project, a set of drivers, a single driver, channels, and policies into a deployed OpenText Identity Manager system in an OpenText eDirectory tree. This can be a production tree or a test tree.

Use the Export feature to make backups of all of your projects and the drivers you want to implement. This way, if something happens to the driver in production, you have a backup.

Use the Deploy feature after you have thoroughly tested the policies that make up your drivers. To test policies, use the Policy Simulator (right-click a policy to see the simulation results of the policy that is being tested) or use the Project Checker to ensure that the project is valid. Then use Deploy to test the policy in a test environment before you deploy the driver into production.

You can also use the Import feature to import an existing OpenText eDirectory driver, a channel, or a policy; after it is imported, you can modify the object or objects, run the Policy Simulator to ensure that the object is working correctly, then deploy the object back into a test tree for further analysis. For more information about policies, see *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

To help you decide on changes to make before deploying, you can use the Compare feature to see differences between the objects you are deploying and those that already reside in an eDirectory tree. See "Using the Compare Feature When Deploying" on page 394.

- "Preparing to Deploy" on page 387
- "Deploying a Project to an Identity Vault" on page 388
- "Deploying a Driver Set to an Identity Vault" on page 389
- "Deploying a Driver to an Identity Vault" on page 390
- "Deploying a Channel to an Identity Vault" on page 392
- "Deploying a Policy to an Identity Vault" on page 393
- "Using the Compare Feature When Deploying" on page 394
- "Troubleshooting Deployed Objects" on page 399
- "Exporting a Project" on page 399
- "Exporting to a File" on page 401

Preparing to Deploy

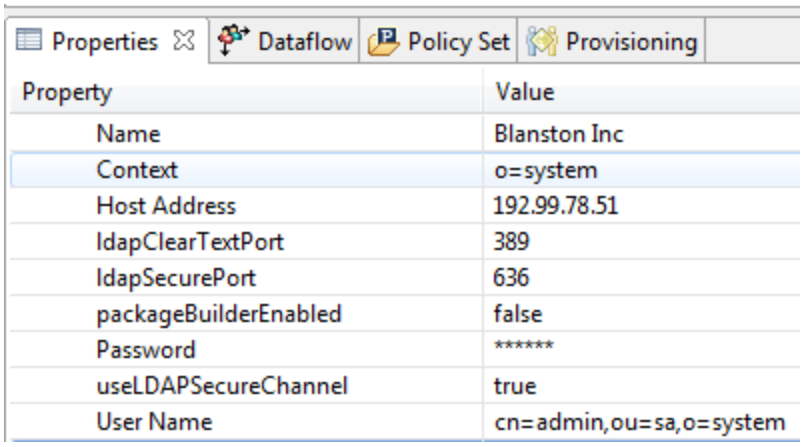
Before deploying a project, run Project Checker and fix any errors that appear.

1 Click **Window > Show View > Project Checker**, then click the **Run the Project Checker** icon.

After you have corrected any problems to the project, make a backup copy of the project before deploying.

Before you deploy objects into an Identity Vault, you need to designate the Deployment DN (distinguished name), or the place in the tree where you plan on deploying the OpenText Identity Manager project or objects.

1 In Designer, select the Identity Vault that contains the object or objects you want to deploy, then look in the Properties view below the Project/Outline view. (You can also open the Identity Vault's or driver's Properties window.)



2 In the Properties view, fill in the information for the Identity Vault if it is not already present.

3 Click the **Browse** icon to find the Deploy Context distinguished name on an existing tree if the other information is accurate and Designer can attach to the tree. You need this information to deploy anything, even a policy.

You can also use the driver set's Deploy Context entry if you want to deploy a driver set to a different context than the one designated in the Identity Vault's Properties view. The driver set's Deploy Context entry overrules the Identity Vault's Deploy Context entry.

IMPORTANT: You must have enough rights to access the OpenText eDirectory tree that is associated with the Identity Vault to which you want to deploy.

Deploying a Project to an Identity Vault

To deploy a project to an OpenText eDirectory tree that is running OpenText Identity Manager, you use the same procedure that you use for deploying a driver set, a driver, channels, or policies. The procedure is described in "Deploying a Driver Set to an Identity Vault" on page 389.

To deploy an OpenText Identity Manager-based project or an object in a project, you must have access to the OpenText eDirectory tree that is associated with the Identity Vault you are designing. You also need to know the deployment DN (distinguished name) context, or the place in the tree where you plan to deploy the OpenText Identity Manager driver set or driver objects.

Deploying a Driver Set to an Identity Vault

Suppose that you finish a new driver set that you want to deploy into a test tree, or suppose that you have imported a driver set, made modifications, and now you want to deploy the driver set back into its working tree. Use the following procedure to deploy an OpenText Identity Manager Driver Set object (and all contained Identity Manager drivers) into an existing Identity Manager system in an OpenText eDirectory tree:

1. Right-click the Driver Set icon in the Modeler view, then click **Live > Deploy**.

   You can also deploy the Driver Set from the Outline view by right-clicking the Driver Set object, then selecting **Live > Deploy**.

   The Identity Vault Credentials window displays if Designer can't authenticate to the eDirectory tree specified in the Identity Vault, or if you do not have the Deployment DN designated in the **Properties** tab of the Identity Vault where you are deploying.

2. Use the Compare feature to see differences between the objects you are deploying and those that already reside in an eDirectory tree.

   See "Using the Compare Feature When Deploying" on page 394.

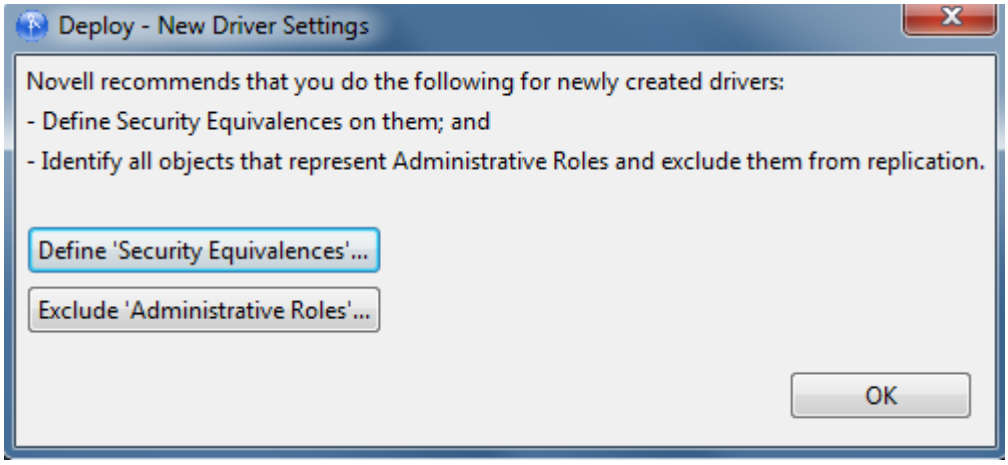
3. In the Deployment Summary window, click **Deploy**.

4. Click **OK** to close the Information window.

5. (Conditional.) If you see other informational messages, decide what action to take.

   You might also see a message in the Deployment Results window stating that the deployment was unsuccessful. Click the error messages in the Operation Results portion of the window to see the error descriptions and possible reasons in the Details portion.

6. (Conditional) If this is a new deployment, the Deploy - New Driver Settings window displays. Define security equivalences on the driver set and identify all objects that represent Administrative roles and exclude them from being replicated.



   In both instances, we recommend that you select the Admin object, and any other objects that qualify in your network environment.

7. Click **OK**.

eDir-to-eDir Deployments and SSL/TLS

By default, always deploy both sides of an eDirectory-to-eDirectory connection when you have SSL and TLS enabled. If SSL/TLS are enabled, Designer creates the certificates in the eDirectory tree when you deploy the drivers. SSL and TLS are not enabled or configured by default.

To check your present SSL settings, click **Window** > **Preferences**, then click **NetIQ** > **OpenText Identity Manager** > **Configuration** and click the **eDir-to-eDir SSL/TLS** tab. After configuration, the Deploy feature uses the SSL preference settings under **Certificate overwrite policy**.

If you changed the default NCP port (524) used for eDirectory-to-eDirectory connection, perform the following actions:

1. Restart the server for the change to take effect.
2. Specify the new port number in the ncpPort attribute in the Identity Vault properties page in Designer.
3. Create the eDirectory-to-eDirectory certificate in Designer.

Deploying a Driver to an Identity Vault

Suppose you finish a new driver object that you want to deploy into a test tree, or suppose you have imported a driver object, made significant modifications, and now you want to deploy that driver object back into its working tree. Use the following procedure to deploy an OpenText Identity Manager Driver object (and all contained channels and policies) into a driver set:

1. Select an Identity Vault in the Modeler view.
2. Right-click a driver object connected to a Driver Set icon in the Identity Vault.

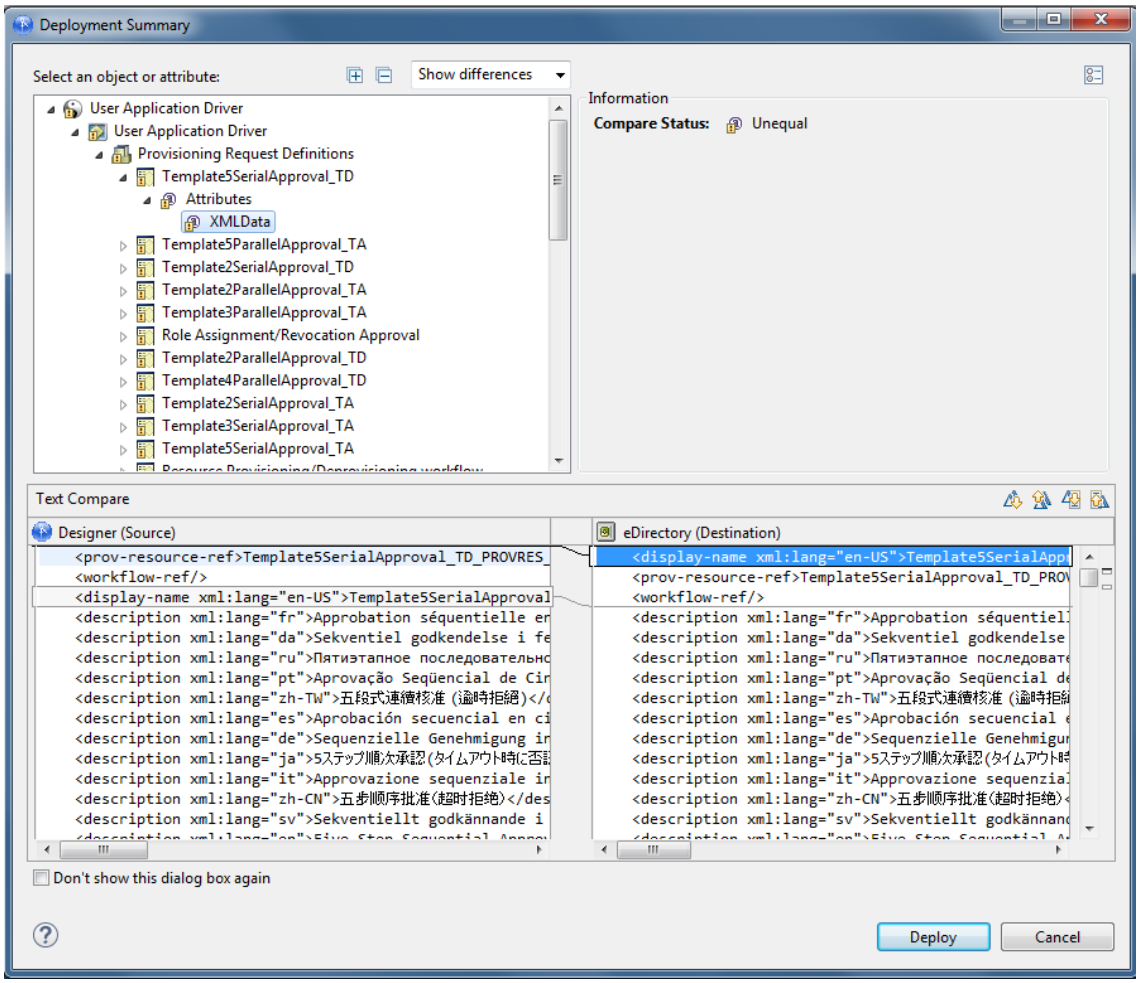
   The driver object is represented by a circle icon.
3. Click **Live > Deploy**.

   You can also select the driver object from the Outline view. Click the **Outline** tab, right-click the driver object you want to deploy, then click **Live > Deploy**.

   NOTE: An error displays if Designer can't authenticate to the OpenText eDirectory tree specified in the Identity Vault, or if you do not have the Deployment DN designated in the **Properties** tab of the Identity Vault you are deploying to.

   ---
4. Review the information displayed in the Deployment Summary window to see the differences between the objects you are deploying and those that already reside in an OpenText eDirectory tree. It is the same as the Compare feature. For more information about how to use the Compare window, see "Using the Compare Feature When Deploying" on page 394.

   When you deploy or reconcile a driver, the OpenText Identity Manager version of the Identity Vault server is updated to match the live system. Updating the OpenText Identity Manager version allows Designer to correctly set the engine controls for the driver so that invalid engine controls are not deployed to the Identity Vault.



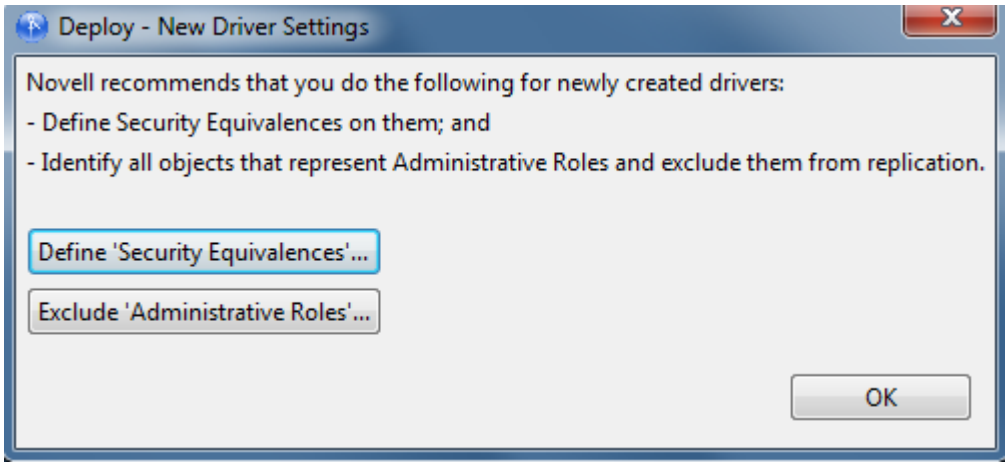
5 Click **Deploy** to begin the process.

6 Click **OK** to close the Deployment Results window.

7 (Conditional.) If you see other informational messages, decide what action to take.

You might see a message in the Deployment Results window stating that the deployment was unsuccessful. Click the error messages in the Operation Results portion of the window to see the error descriptions and possible reasons in the Details portion.

8 (Conditional) If this is a new deployment, the Deploy - New Driver Settings window displays. Define security equivalences on the driver set and identify all objects that represent Administrative roles and exclude them from being replicated.



In both instances, we recommend that you select the Admin object, and any other objects that qualify in your network settings.

You can modify security equivalences and excluded roles after the driver is deployed. To do so, right-click the driver object and select **Live** > **Set Up Driver Security**, or right-click the Application object and select **Driver** > **Set Up Driver Security**.

9 (Conditional) If this is a new deployment and the driver has a job associated with it, the Deploy - Security Equivalences dialog is displayed. This dialog is not displayed for the subsequent deployments.

To add or delete security equivalences, click **Add** or **Delete**.

Deploying a Channel to an Identity Vault

A channel is a grouping of rules and policies, and Designer allows you to deploy a channel object into a driver if necessary. The Subscriber and Publisher channels describe the direction in which the information flows. The Subscriber channel takes the event from Identity Vault (OpenText eDirectory) and sends that event to the managed system (application, database, CSV file, etc). The Publisher channel takes the event from the application, database, CSV file, etc., and sends that event to the Identity Vault. The Subscriber and Publisher channels act independently; actions in one are not affected by what happens in the other.

Channel objects must be a part of a newly created driver, or they must be a part of an existing driver that now needs to be modified. Driver objects are created through the Designer or iManager utilities. Because channel objects are a part of a driver object, you deploy a channel object into an existing driver object. If you simply deploy the channel object, Designer creates a skeleton driver as a placeholder for the channel object.

To deploy an OpenText Identity Manager channel (a Subscriber channel or a Publisher channel) object and all contained policies into a driver in an Identity Vault:

1 In the **Outline** tab, select the channel object under the driver object. The driver object is represented by a circle icon; the Publisher icon shows a black dot on the icon and the Subscriber icon shows a white dot .

2 Right-click the channel object you want to deploy, then click **Live > Deploy**.

An error displays if Designer can't authenticate to the OpenText eDirectory tree specified in the Identity Vault, or if you do not have the Deployment DN designated in the **Properties** tab of the Identity Vault you are deploying to.

The Deployment Summary window shows you the differences between the objects you are deploying and those that already reside in an OpenText eDirectory tree. It is the same window format as the Compare feature. For more information about how to use the Compare window, see "Using the Compare Feature When Deploying" on page 394.

An error displays if Designer can't authenticate to the OpenText eDirectory tree specified in the Identity Vault, or if you do not have the Deployment DN designated in the **Properties** tab of the Identity Vault to which you are deploying.

3. In the Deployment Summary window, click **Deploy**.
4. After the channel deploys, click **OK** to close the Deployment Results window.

Deploying a Policy to an Identity Vault

A policy is a collection of rules and arguments that allow you to configure an application so it can send and receive events between itself and an Identity Vault (OpenText eDirectory). You use policies to manipulate the data you receive from an Identity Vault or from the application. Each driver performs different tasks and policies tell the driver how to manipulate the data to perform those tasks. For more information about policies, see *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

To deploy an OpenText Identity Manager Policy object (for example, a rule or a style sheet) into a driver or channel (Subscriber or Publisher):

1. Click the **Outline** tab and select a policy under a driver object or a channel object.

   Policies can be of the type DirXML Script, Schema Mapping, or XSLT style sheet, and each type has its own icon.
2. Right-click a policy object, then select **Live > Deploy**.

   An error displays if Designer can't authenticate to the OpenText eDirectory tree specified in the Identity Vault, or if you do not have the Deployment DN designated in the **Properties** tab of the Identity Vault you are deploying to.

   The Deployment Summary window shows you the differences between the objects you are deploying and those that already reside in an OpenText eDirectory tree. It is the same window format as the Compare feature. For more information about how to use the Compare window, see "Using the Compare Feature When Deploying" on page 394.
3. In the Deployment Summary window, click **Deploy**.
4. After the policy deploys, click **OK** to close the Deployment Results window.

Using the Compare Feature When Deploying

The Compare feature allows you to see differences between driver sets, drivers, channels, and policies that are stored in projects and those that are running in deployed systems. Previous versions of Designer only provided conflict resolution when importing a driver. While importing, you could select which policies of the driver you wanted to update, but you could not view any differences between existing and new values.

Designer now provides conflict resolution on an object-by-object basis and allows you to view the differences between existing and new values when importing and deploying driver sets, drivers, channels and policies. For example, before deploying a driver object in Designer to a driver object that already exists in the Identity Vault, you can run Compare. Compare shows whether the driver objects are equal (no action is necessary) or unequal. If they are unequal, you can choose not to reconcile the driver objects, choose to update the driver object in Designer, or choose to update the driver object in OpenText eDirectory.

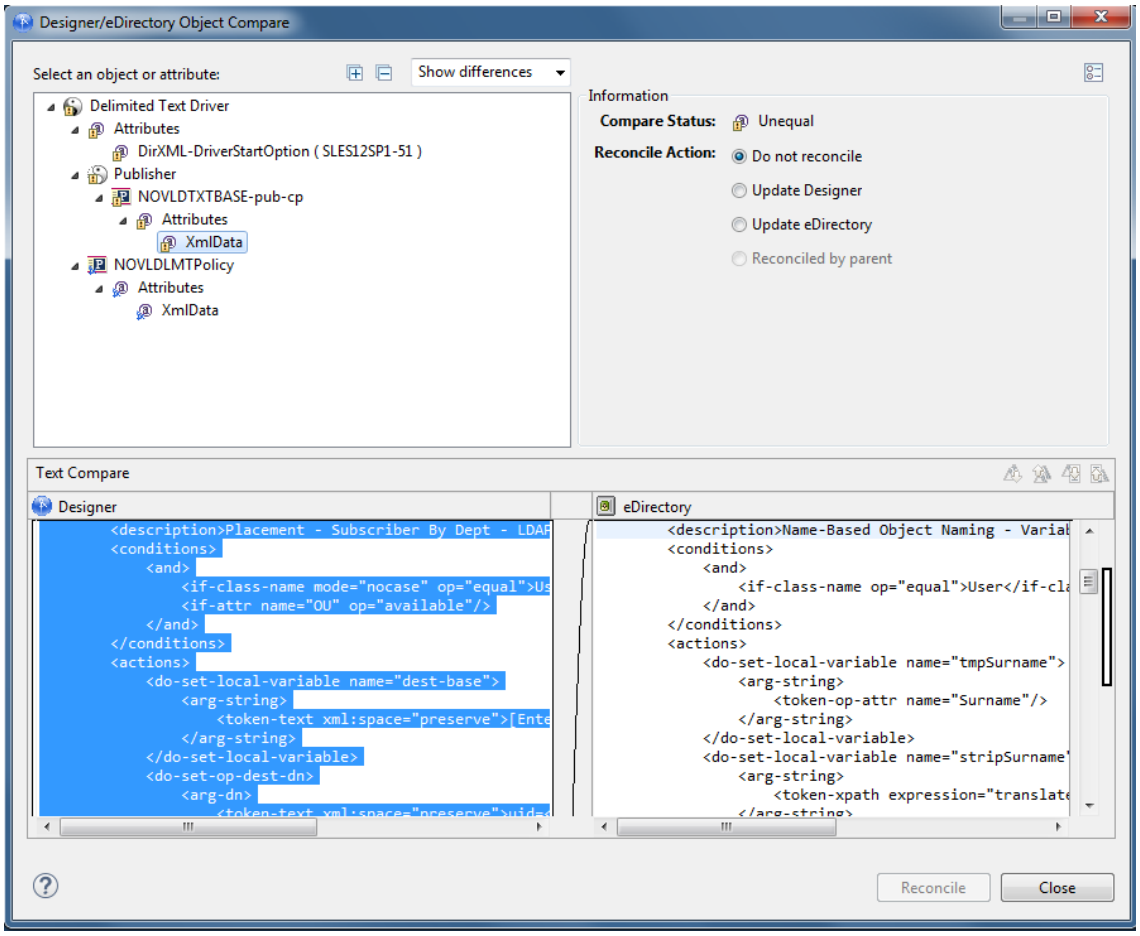
You can run the Compare feature at any time. If you choose to reconcile the differences between drivers objects in Designer and OpenText eDirectory while in Compare, you won't need to separately run Import or Deploy to make the changes.

- "Using Compare when Deploying a Driver Object" on page 394
- "Using Compare Before Deploying a Channel Object" on page 397
- "Using Compare Before Deploying a Policy" on page 397
- "Matching Attributes with Designer Properties" on page 397
- "Comparing Driver Set and Driver Attributes" on page 398
- "Renaming and Deleting Deployed Objects" on page 398

Using Compare when Deploying a Driver Object

Suppose you want to determine if you have deployed all of the changes you have made to a driver object in Designer to the same driver in the Identity Vault.

1 Right-click the driver object in either the Modeler view or in the Outline view. Select **Live > Compare** to bring up the Designer/OpenText eDirectory Object Compare window.



2 In the **Select an object or attribute** portion of the window, you see the listed objects and attributes. Select the attributes and child objects to see the actual differences displayed in the **Text Compare** portion of the window.

   The plus icon at the right side of the **Select an object or attribute** allows you to expand all elements in the parent object, and the minus icon collapses all of the elements. The ? icon displays the Summary/Compare dialog box help. Server-specific attributes, which are attributes that have a value for each server that is associated with a driver set, are displayed in the **Attributes** list with the server name in parentheses to the right of the attribute name.

3 By default, the Compare window only displays values that are different between OpenText eDirectory and Designer. To view all of the object values, select **Show all** from the pull-down menu. Your choices are **Show differences**, **Show deletes**, and **Show all**.

4 Check to see the status of the values that are shown.

   Values that are equal are shown as **Equal** on the **Compare Status** line in the Information portion of the Compare window.

   The overlay image displayed in the **Compare Status** entry identifies objects or attributes that need reconciliation. The following table describes what you see in the **Compare Status** line and the overlays that you can see:

Compare Status	Description
Equal	The selected attribute's value or all attributes of the selected object are the same in OpenText eDirectory and Designer.
Unequal 	The value of the selected attribute, or one or more attributes of the selected object, are different in OpenText eDirectory and Designer.
Not Deployed 	The selected object or the object containing the selected attribute is not deployed to OpenText eDirectory.
Not Imported	The selected object or object containing the selected attribute does not exist in Designer or when the object is referenced by one or more objects being compared.
Renamed	Designer tracks objects that are deployed, then renamed in the Designer project. The Designer and OpenText eDirectory DNs are displayed in the value fields.
Unknown	The selected object or object containing the selected attribute cannot be compared, such as a password.
Deleted	Designer also tracks objects that are deployed, then deleted from the Designer project. To delete the object from OpenText eDirectory during deployment, select **Delete the Identity Vault object**.

You can also see an **Attribute Note** if you select an attribute.

5 Under the Information portion of the Compare window, select how you want to reconcile the differences between the Source and Destination.

If **Compare Status** shows **Unequal**, you have three choices:

- To do nothing, keep the default value of **Do Not Reconcile**.
- To update the driver in Designer so that it contains the same information as the driver in OpenText eDirectory, select **Update Designer**.
- To update the driver in OpenText eDirectory to reflect the changes you have just made to the driver in Designer, select **Update eDirectory**.

The green check box in the bottom corner of the icons shows all of the child objects that are being reconciled with the parent object. If you select the parent object to perform the update, then all of the child objects under the parent reflect that choice and you see the **Reconciled By Parent** icon selected. If you do not choose a parent object, you can reconcile each child object individually. You can also see a small Designer icon and an OpenText eDirectory icon, showing how objects are being reconciled.

6 Check to see the **Text Compare** values.

The **Text Compare** values displayed in the bottom portion of the Designer/OpenText eDirectory Object Compare window vary, depending on the object being compared. For instance, Compare shows changes to policies or XML data. The Text Compare dialog box uses the Eclipse Compare editor to compare attributes that contain XML data, such as policy data, driver filters, or configuration data. The differences in the code are highlighted in blue.

7 After you view the differences, click **Reconcile** to perform the reconciliation actions for each object in the tree, or click **Close** to close the Designer/OpenText eDirectory Object Compare window.

Using Compare Before Deploying a Channel Object

Suppose you want to deploy a channel object from the Identity Vault and the same channel already exists in Designer. You can compare the two channels to see similarities and differences.

1 Right-click the channel object in the Outline view.

2 Select **Live > Compare** to bring up the Designer/OpenText eDirectory Object Compare window.

All Compare windows behave the same as described in "Using Compare when Deploying a Driver Object" on page 394.

Using Compare Before Deploying a Policy

Suppose you want to deploy a policy object from the Identity Vault and the same policy already exists in Designer. You can compare the two policies to see similarities and differences

1 Right-click the policy object in the Outline view.

2 Select **Live > Compare** to bring up the Designer/OpenText eDirectory Object Compare window.

All Compare windows behave the same as described in "Using Compare when Deploying a Driver Object" on page 394.

Matching Attributes with Designer Properties

The attributes of the object are displayed in the single select attribute list. Selecting an attribute displays its value below the attribute list with the Designer value on the left and the OpenText eDirectory value on the right. The name displayed in the list is the OpenText eDirectory attribute name.

Three tables map the attribute to the Designer property page or control, where you can change or set the attribute (you can't make changes inside the Compare window). Table 11-1 on page 319 shows driver set attributes, Table 11-2 on page 319 shows driver attributes, and Table 11-3 on page 320 shows channel attributes.

Comparing Driver Set and Driver Attributes

Use the Compare feature to compare the attributes of a driver set or a driver without comparing all of the child objects.

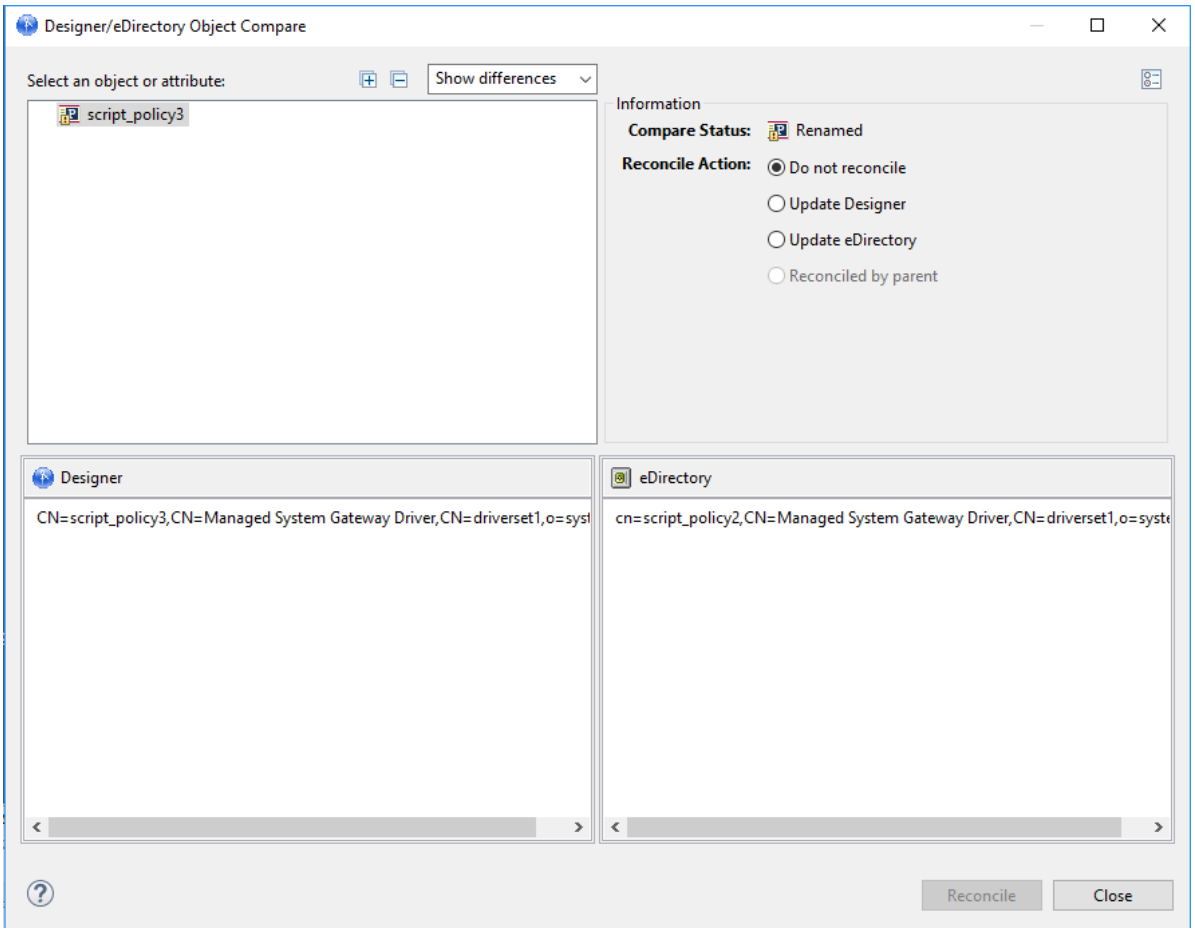
1 Right-click the driver set or driver, then select **Live > Driver Set Configuration > Compare Attributes**.

By default, the Compare windows shows only those attributes that are unequal, but you can select to show deletes, or show all attributes.

Renaming and Deleting Deployed Objects

Designer tracks objects that are deployed, then renamed in the Designer project. The Designer and OpenText eDirectory DNs are displayed in the value fields. The renamed objects are displayed in the Deployment Summary window and the Compare Status entry displays **Renamed**.

Figure 15-1 *Renamed Drivers and Driver Sets*



During the deploy operation, the renamed Designer object is renamed in OpenText eDirectory. When performing a compare operation, you can reconcile the object by updating either the Designer or OpenText eDirectory object name. Only objects that are renamed in Designer are tracked. If an object is renamed in OpenText eDirectory, Designer might not locate the associated object when building the compare summary.

Designer also tracks objects that are deployed, then deleted from the Designer project. When you deploy the parent of the object that is deleted, you are given the option to delete the object from the Identity Vault. To delete the object from OpenText eDirectory during deployment, select **Delete the Identity Vault object**. You can select **Show deletes** from the drop-down menu.

Designer removes the object from the deleted object list if the parent is deployed and the object is not marked for deletion. In the following graphic, a driver was deleted from the driver set.

You can use the Compare feature to delete a deleted object from OpenText eDirectory or you can re-import the object into Designer.

For example, to delete the object from OpenText eDirectory, select **Update eDirectory** from the **Reconcile Action** selection. To re-import the object into Designer, select **Update Designer**. Only objects that are deleted in Designer are tracked. If an object is deleted in OpenText eDirectory, Designer shows the object as not deployed and creates a new object when you run Deploy or Compare.

Troubleshooting Deployed Objects

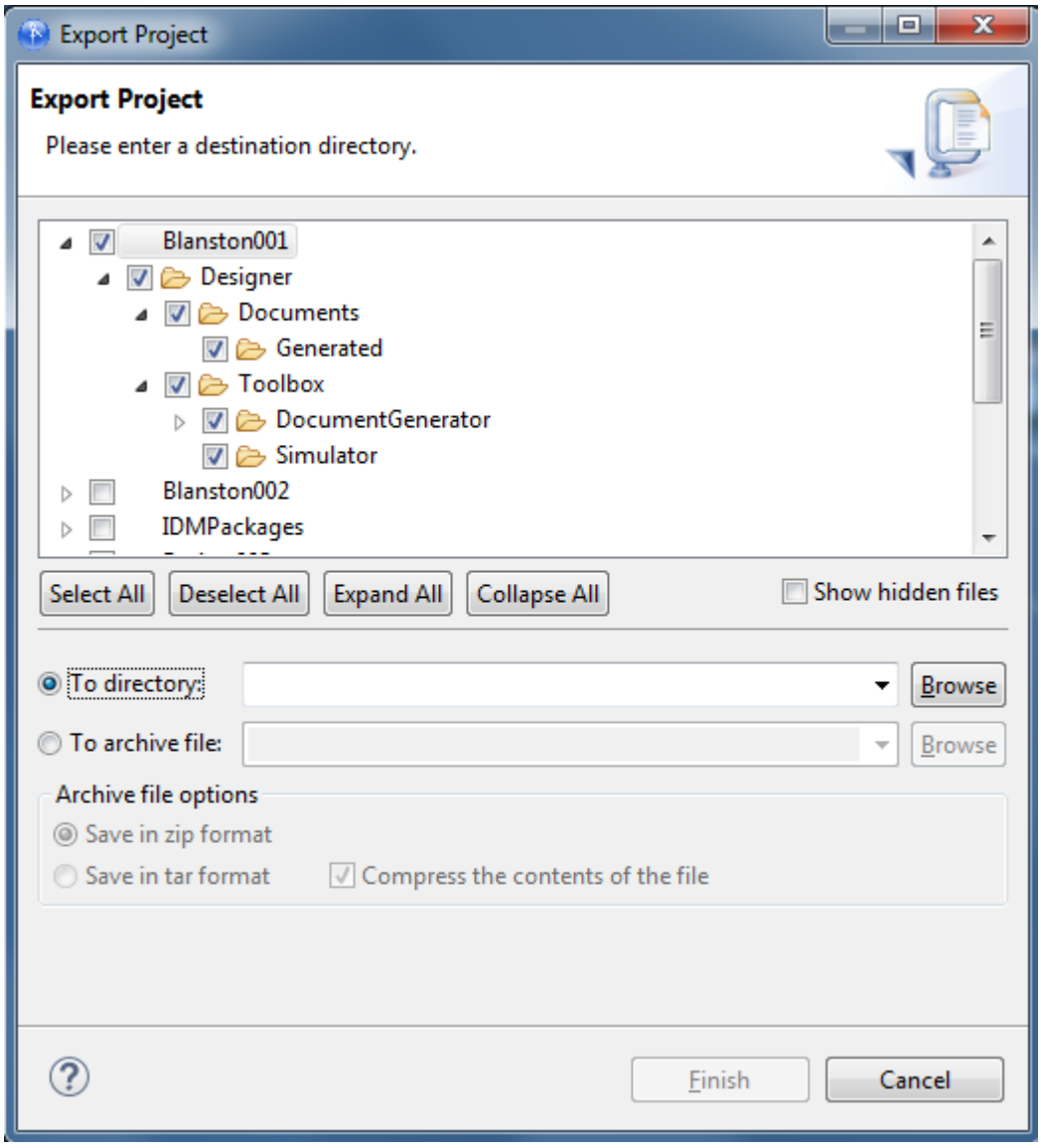
For information about troubleshooting deployed objects, see "Deploying OpenText Identity Manager Objects" on page 536.

Exporting a Project

The Export feature allows you to export Projects and Driver Configuration files to a local, removable, or network directory.

1. Click **File > Export**.

   You use the Export window to export an existing OpenText Identity Manager Project to an archive file or to an iManager configuration file.
2. Select **Designer for OpenText Identity Manager > Export Designer Project**, then click **Next**.
3. In the Export File System window, select the projects you want to export.



4 Click **Select All** to select all projects in the `designer_workspace` directory (for Windows, the default location is `C:\Documents and Settings\user's_login_name\designer_workspace`).

or

Click **Deselect All** to clear the selections. You can then select the projects you want to export. Use the **Expand All** or **Collapse All** icons to expand or collapse the objects under each project. You can also select **Show hidden files** to display any files that have a period (.) at the beginning of the filename.

IMPORTANT: You must select all items relating to a project for an export of the project to work.

You can also browse to the directory location where you want to select the resources.

5 After you designate the directory to which to export the projects, click **Finish**.

You can also export projects to an archive file:

1. Click **File > Export**.

   You use the Export window to export an existing OpenText Identity Manager Project to an archive file or to an iManager configuration file.
2. Select **Designer for OpenText Identity Manager > Export Designer Project**, then click **Next**.
3. Select the **To archive file** option in the Export window.
4. Select the projects you want to archive.
5. Designate where you want the archive file saved. You can browse to an already existing file, or type an archive filename.
6. Select the archive format (zip or tar).
7. Select whether you want to compress the contents of the file, then click **Finish**.

   With the Project Export Wizard, you don't need to select the model files that are necessary for the project to work, because these files are exported automatically. You can choose to not export any extra files that are included in a project by deselecting them under the project in the Export Project window.

Exporting to a File

You can use the export feature to export everything you create in Designer, from projects containing all Identity Vaults and their driver sets down to a single policy. If you export a driver configuration file that contains only a policy, Designer creates the parent containment objects, such as a channel, a driver, or a driver set, as part of the exported policy object. These parent containment objects do not contain attributes; they are only the framework of the channel, driver, or driver set.

The exported `.xml` files are compatible with those used by the iManager driver configuration file plug-ins for Identity Manager 2.0.2 and above. This allows you to export configuration files from Designer and import those files through iManager or through Designer's import feature.

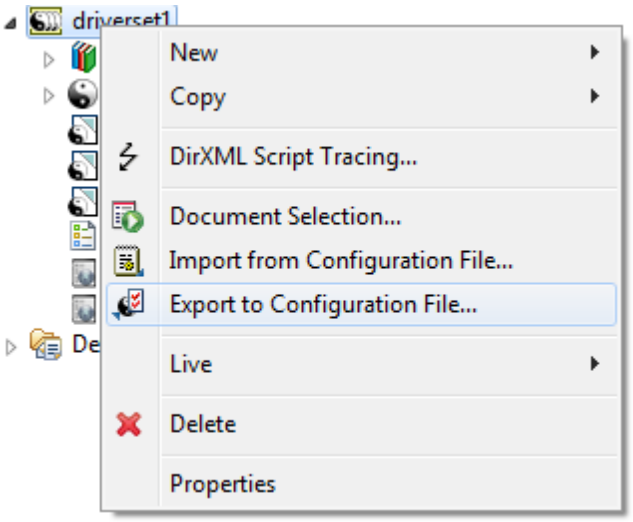
You can export a driver configuration to a file from a number of places, including:

- "Using the Export Context Menu" on page 401
- "Exporting Configuration Files from the Modeler View" on page 402
- "Exporting Configuration Files from the Outline View" on page 402

Using the Export Context Menu

To export a driver set and all of the associated objects such as drivers, channels, and policies:

1. Right-click the driver set in the Outline View.
2. Select **Export to Configuration File**.



Designer uses the name of the driver sets for the `.xml` file.

3 For future reference, name each driver set to denote that it is a driver set and denote the Identity Vault it comes from. You can also add a date to the name.

4 Click **Save**.

5 To close the Export Configuration Results window, click **OK**.

Exporting Configuration Files from the Modeler View

1 Double-click the **System Model** icon under a project name in the Project view to open the project model in the Modeler view.

2 Right-click the Driver Set object inside an Identity Vault icon, then select **Export to Configuration File**.

3 In the Export Driver Configuration window, select a filename and location to use in future references. You can also add a date to the filename if you save a lot of driver iterations.

By default, Designer uses the name of the driver or driver set corresponding to the object selected. If you right-click an Identity Vault or Driver Set object, you see the Driver Set name in the File Name entry. If you have more than one Driver Set object in the Identity Vault, you see the Export Driver Configuration window with the name of that driver set in the File Name entry for each Driver Set object.

4 Select the directory where you want to store the file, then click **Save**.

Exporting Configuration Files from the Outline View

You can use the Outline View to save driver sets, drivers, channels, and policy configuration files to local, removable, or network directories. The following procedure documents steps for exporting channels and policies.

1 Double-click the **System Model** icon under a project name in the Project view. This brings up the project in the Modeler view.

2 Click the **Outline** tab.

3 Right-click a channel object under a driver object, then select **Export to Configuration File**.

4 From the Export Driver Configuration window, select a filename and location to use in future references. You can also add a date if you are backing up multiple iterations of the file.

   By default, Designer uses the name of the driver or object corresponding to the object selected. You might also need to designate that it is the Publisher channel of an Active Directory driver, along with the date when you saved the file.

5 Click **Save**.

6 In the Export Configuration Results window, click **OK**.

To export one or more policies from a driver or channel:

1 From the Outline view, right-click a Policy object and select **Export to Configuration File**.

   You can also use the Ctrl key to select more than one policy, then right-click them as a group and select **Export Policy to Configuration File**.

2 From the Export Driver Configuration window, select a filename and location to use for future reference. You can also add a date if you are backing up multiple iterations of the file. If you are exporting policy files from multiple drivers, include driver and channel information in the filename.

3 Click **Save** for each policy selected.

   Each policy is saved to its own `.xml` file. By default, Designer uses the name of the policy or rule selected.

4 In the Export Configuration Results window, click **OK**.

16 The NetIQ XML Editor

This section provides an overview of the features of the NetIQ XML Editor.

- "About the NetIQ XML Editor" on page 405
- "Using the Source Editor" on page 411
- "Using the Tree Editor" on page 413
- "Attaching a Schema or DTD" on page 415
- "Setting XML Editor Preferences" on page 416

About the NetIQ XML Editor

The NetIQ XML editor lets you create, edit, and validate XML files.You can edit XML files in either the Source or Tree editor. You can customize the certain behaviors, such as code completion, on the **Preferences** tab.

The NetIQ XML editor is built on the Web Standard Tools (WST) project architecture.

Creating XML Files

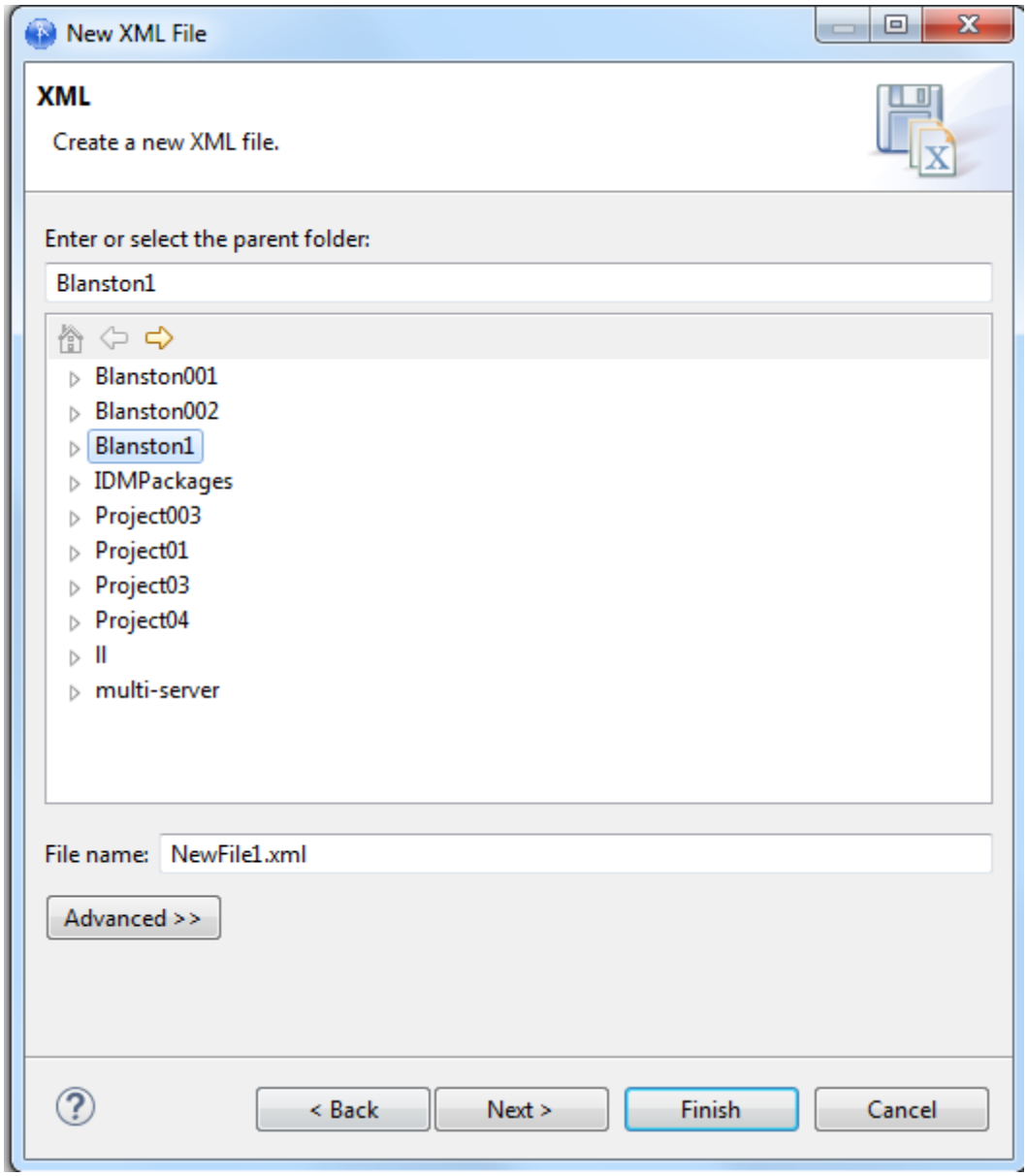
You use the New XML File Wizard to create new XML files. The wizard can create an empty XML file or a generated XML file based on an XML schema or DTD. Generated files contain skeleton XML data that is based on a given root element and an XML schema or DTD.

To launch the New XML File Wizard:

1. Click **File > New > Other.**
2. Select **Show All Wizards**.
3. Expand the XML Folder, select **XML**, then click **Next**.
4. (Optional) If Designer asks you to enable a particular activity, click **OK**.
5. Fill in the fields as follows:

Field	Description
Enter or select parent folder	Specify where the wizard should create the new file.
File name	Specify the name of the new file.
Advanced >>	Click this button if you want to specify that the new XML file should link to another file in the file system.

6. Specify the name and location for the new file and click **Next**.



7 Choose one of the source options on which you want to base the new XML file.

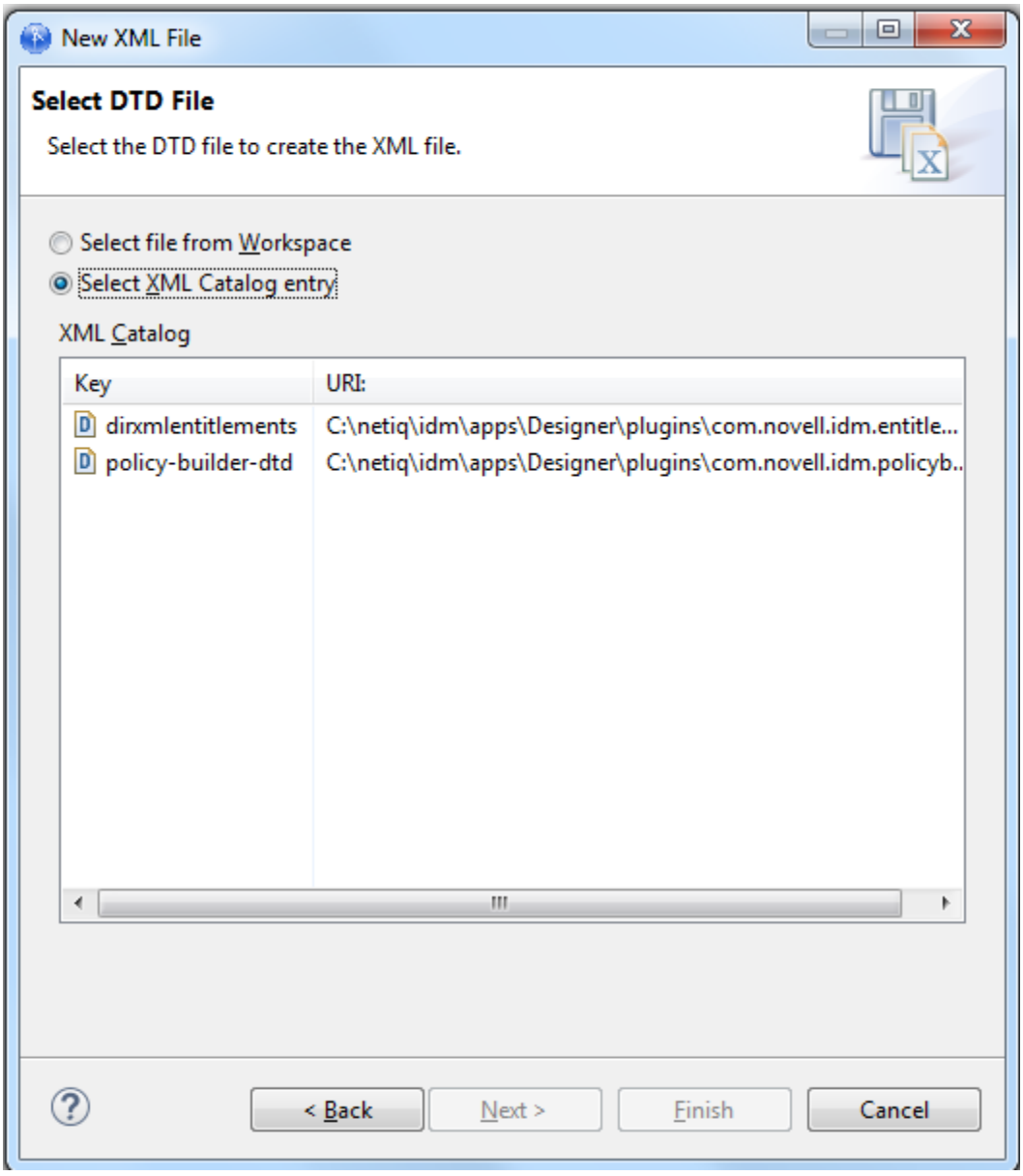
Option	Description
Create XML file from a DTD file	Generates an XML document containing a root element and a skeleton based on a DTD that you either import or choose from an existing catalog entry.
Create XML File from an XML schema file	Generates a skeleton XML document containing a root element and skeleton based on a schema that you either import or choose from an existing catalog entry.
Create XML File from an XML template	Creates an XML document containing the XML declaration with the version and encoding attributes set to 1.0 and UTF-8 by default.

8 Click Next.

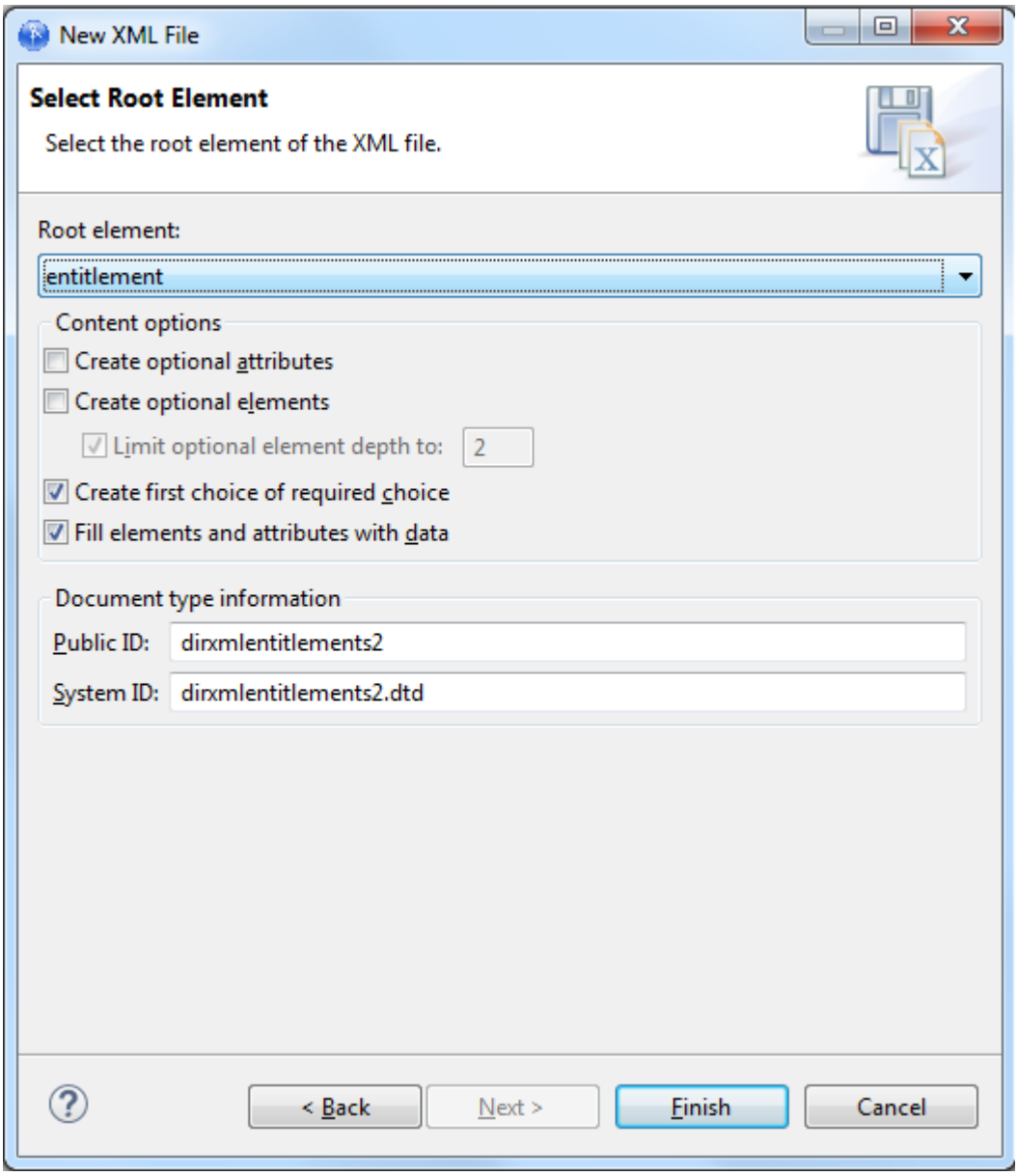
9 (Conditional) If you selected Create XML File from a DTD file or Create XML File from an XML schema file, complete the following steps:

9a Choose one of the following fields:

Field	Description
Select file from Workspace	If you choose this option, you must select from a list of DTDs or schemas in your workspace. You can also choose to import a new schema into your project if the schema is not available.
Select XML Catalog entry	Choose one of the XML Catalog entries from the list. You can edit this list in Preferences > Web and XML > XML Catalogs.



9b Click **Next**. You are prompted to specify the root element.



9c Fill in the fields as follows:

Root element: Choose or type the new document's root element.

Create optional attributes: Select this option if you want the wizard to generate optional attributes.

Create optional elements: Select this option if you want the wizard to generate optional elements.

Create first choice of required choice: Select this option if you want the skeleton XML to always contain the first choice in a required choice. If this is not selected, no elements are inserted for the choice.

Fill elements and attributes with data: Select this option if you want the wizard to generate dummy data in the file for elements and attributes.

The generated XML inserts the node name as the data of the elements

Public ID: Specify the file's Public ID

System ID: Specify the file's System ID.

10 (Conditional) If you selected **Create XML File from an XML template**, select the template you want to use or clear **Use XML Template** to create an empty XML file.

11 Click **Finish**.

Validating Files

You can validate your XML files by using the right-mouse menu in the Source editor. If any validation errors or warnings occur, they are displayed in the Problems view.

Outline View

The XML editor provides an Outline view containing a tree that displays the structure of the XML document including its nodes, elements, attributes, text nodes, comments, and so on from the document.

The Outline tree is closely connected to the Source editor and the Tree editor. When you edit a document in either the Source or Tree editor, the Outline tree updates automatically. If the editing results in a document that isn't well-formed, the structure displayed in the tree might seem odd. But the structure corresponds as closely as possible to the well-formed parts of the document.

Editing or generally moving the cursor in the Source editor or changing the selection in the Tree editor expands and selects the corresponding node (if possible) in the Outline tree. This makes it possible to easily locate the current place in the document.

In a similar fashion, selecting a node in the Outline tree moves the cursor in the Source editor to the textual position of the node (if the Source editor is active) or changes the selection in the Tree editor (if the Tree editor is active). The Outline view provides structural editing capabilities such as inserting and removing nodes.

XPath Navigator

The XPath Navigator view supports syntax highlighting and context-sensitive editing of XPath expressions. It automatically attaches to the currently selected XML editor and uses its Document node as the evaluation context. The namespace context shows all namespaces in scope on its document element.

The view consists of two parts—an editor pane and a results table. When the user types an expression in the editor pane and pauses for 0.5 seconds, the result is shown in the table. If the result is a node list, each row in the table displays an icon for the node type, a short description of the node, and the location of the node in the text (line numbers). Selecting a row in the table selects the text of the corresponding node in the XML editor. However, this is only supported in the Source editor.

Typing Ctrl-Space, '/', '[' or '(' triggers code-completion—the expression is evaluated up until the cursor location, and insertable elements are shown in a drop-down box.

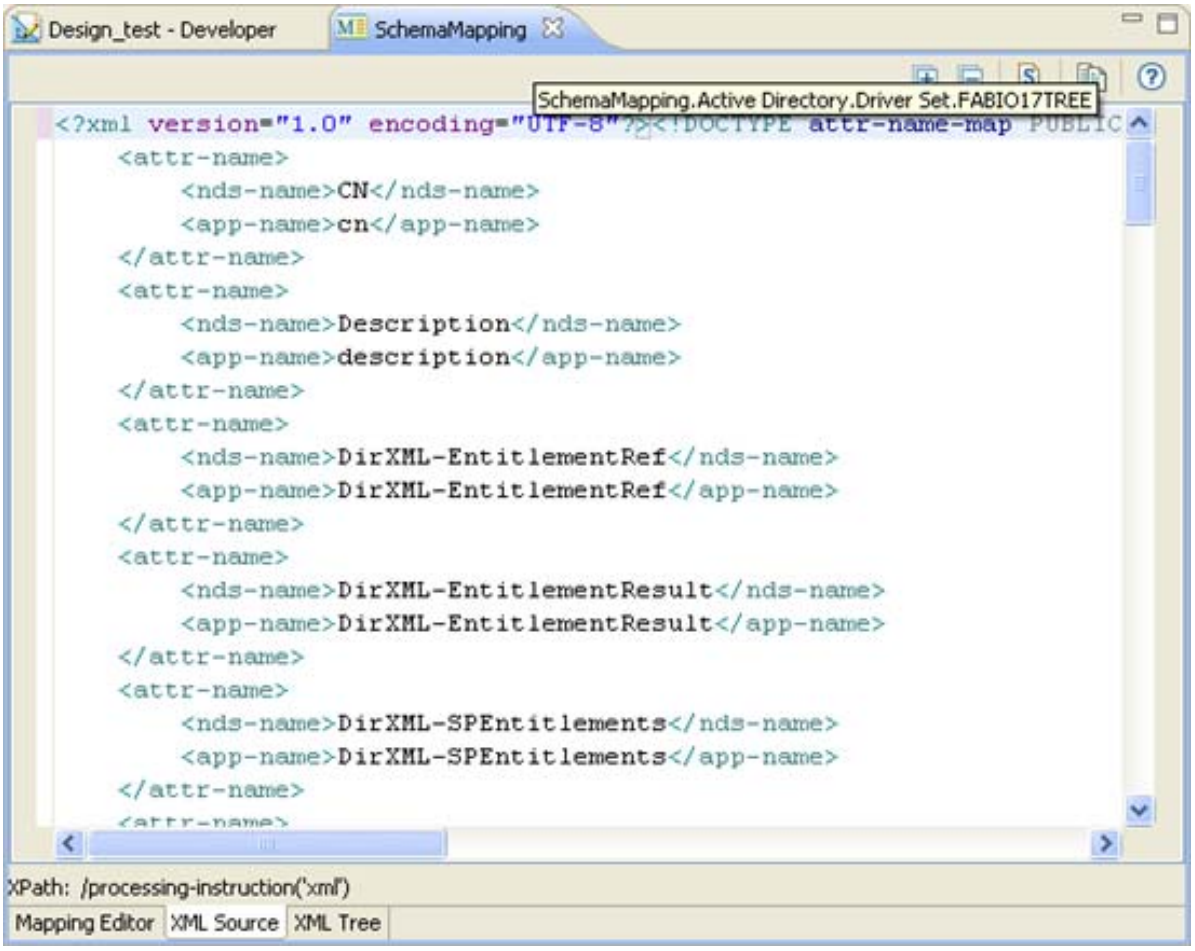
Using the Source Editor

The Source editor supports the following features:

- Syntax highlighting.
- Context-sensitive code completion based on the DTD and the XML schema.

  The code completion is based on the existing content of the XML document if no DTD or XML schema is associated with the XML document. When code completion is activated and the XML document contains `<root><a><b/></a><a></a></root>`, then you type the second <a>, the editor suggests that you add b as a child of the a element.
- As-you-type validation. If the XML is invalid (for example, the > is removed from a tag), the editor indicates the error.
- General text editing operations such as undo, redo, cut, copy, paste, select all.

Figure 16-1 *XML Source Editor*



The XML Source editor provides the following toolbar options:

Table 16-1 *XML Editor Toolbar*

Icon	Description
	Expands all folding (if code folding is enabled). You can enable or disable code folding in two ways; ◆ Selecting **Windows > Preferences > General > Editors**. Select **Structured Editors**. Select **Enable Code Folding**. ◆ In the Source editor, right-click in the left ruler to access the **Folding** submenu.
	Collapses all folding (if code folding is enabled)
	Attaches a schema. For more information about using this feature, see "Attaching a Schema or DTD" on page 415.
	Shows help

The Source editor right-click menu contains these options:

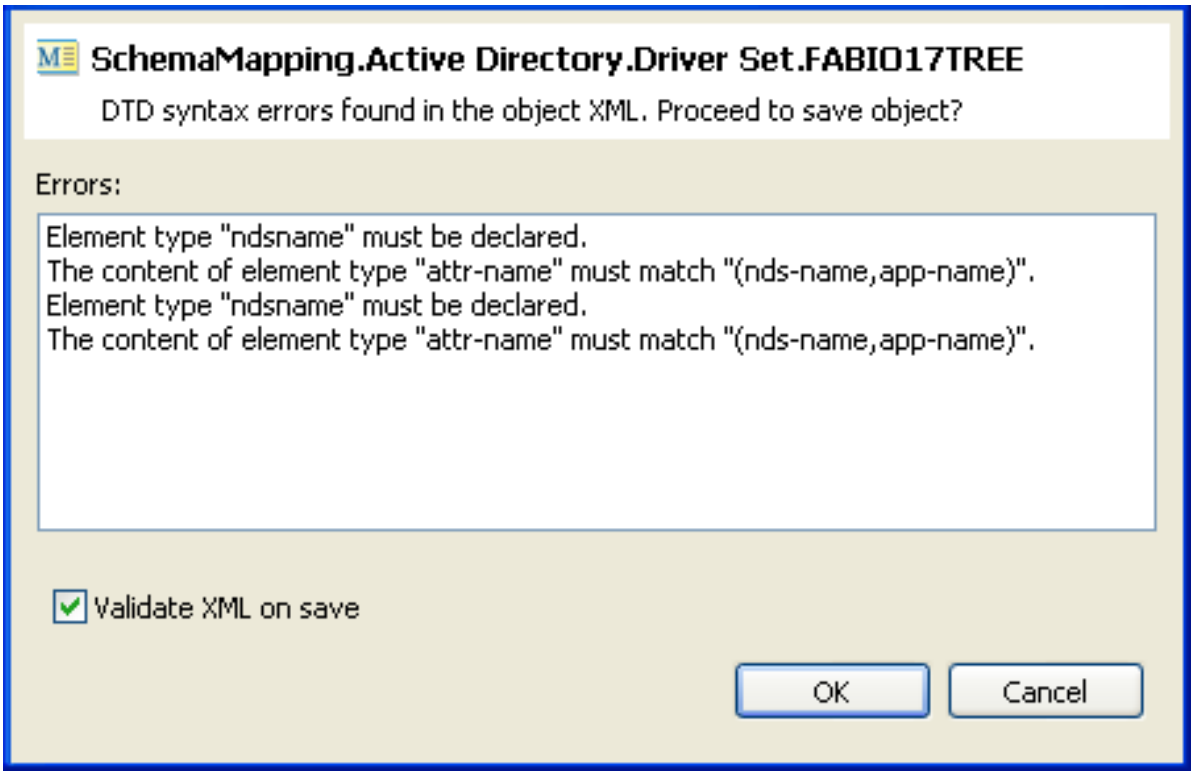
Table 16-2 *XML Source Editor Right-Click Menu Options*

Menu Choice	Description
Revert File	Removes any changes to the XML file.
Cut, Copy, Paste, Undo, Save	Performs the common editor function.
Format	**Document:** Formats the entire document as specified in the preferences. **Active elements:** Formats only selected elements.
Clear Validation Errors	Clears reported validation errors from the Problems view.
Validate	Validates the XML document and shows errors in the Problems view.
Preferences	This is the same as setting preferences by using the **Windows > Preferences** option. For more information, see "Setting XML Editor Preferences" on page 416.

To save XML updates, do one of the following:

- Click **Save**  in the Designer toolbar.
- Right-click in the XML editor, then select **Save**.
- Press Ctrl+S.

When saving, the XML editor automatically checks the XML to make sure it conforms to the appropriate DTD (Filter DTD, DirXML Script DTD, etc.) It saves non-conforming XML only if you explicitly instruct it to do so. For information about OpenText Identity Manager DTDs, see the *OpenText Identity Manager DTD Reference* (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html).



NOTE: You can disable notification of DTD errors in Designer Preferences. To do so, select **Window** > **Preferences**, then select **NetIQ** > **OpenText Identity Manager** > **Configuration** in the left navigation. Deselect **Prompt for errors when validating XML against DTD for all Policy Editors**.

Using the Tree Editor

The Tree editor supports these features:

Direct Editing: You can directly edit the text fields, including element names, attribute names and values, namespace names and values, text, and comments.

Insertion: You can insert new nodes by using the Tree editor's right-click menu, which allows you to insert nodes as children before or after the selected node. If the node is an element, you can insert attributes. The submenus for **Add Child**, **Add After**, **Add Before** contain the node that can be legally added. If no schema or DTD is associated with the document, the submenus contain **New Attribute** or **New Element**.

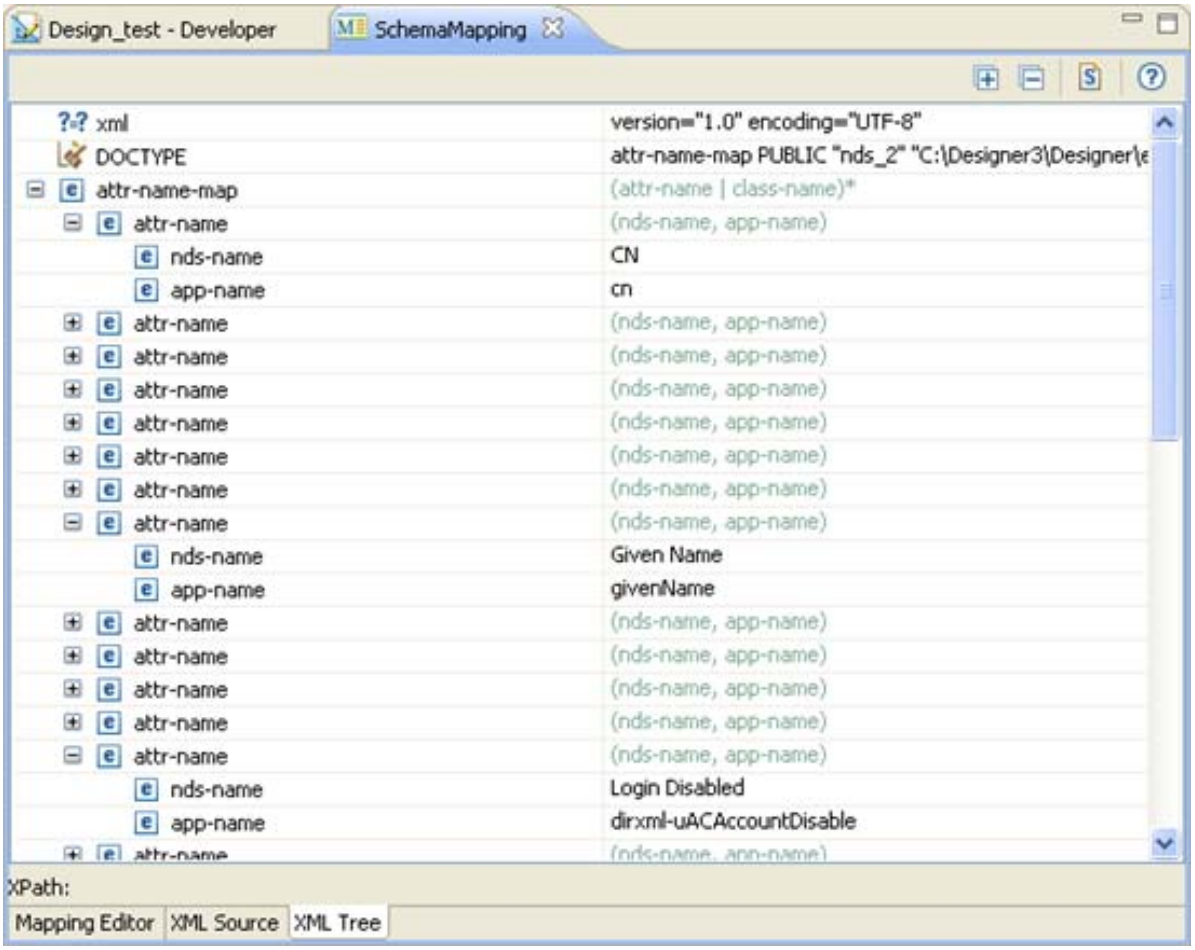
Deletion: To delete a node, select it and either press the Delete key or right-click, then click **Remove**.

Drag-and-drop: You can use this functionality inside the tree and between trees.

General Editing: You can perform operations such as undo, redo, cut, copy, and paste.

The Tree editor displays the XML nodes, with the value of each node displayed in a table cell next to the tree node.

Figure 16-2 *XML Tree Editor*



The Tree editor provides the following toolbar options:

Table 16-3 *Tree Editor Toolbar*

Icon	Description
	Expands all nodes.
	Collapses all nodes.
	Attaches a schema. For more information about using this feature, see "Attaching a Schema or DTD" on page 415.
	Launches help.

Attaching a Schema or DTD

Both the Source editor and Tree editor allow you to attach an XML schema or DTD from the toolbar.

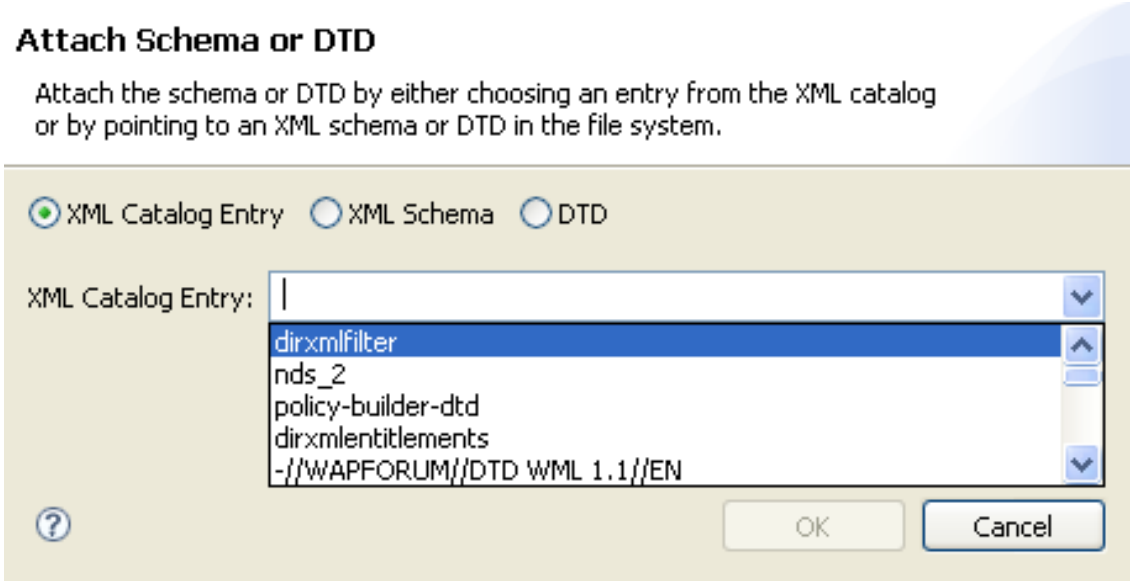
1 In the XML Source or XML Tree editor, click Attach .

This opens the Attache Schema or DTD dialog box.

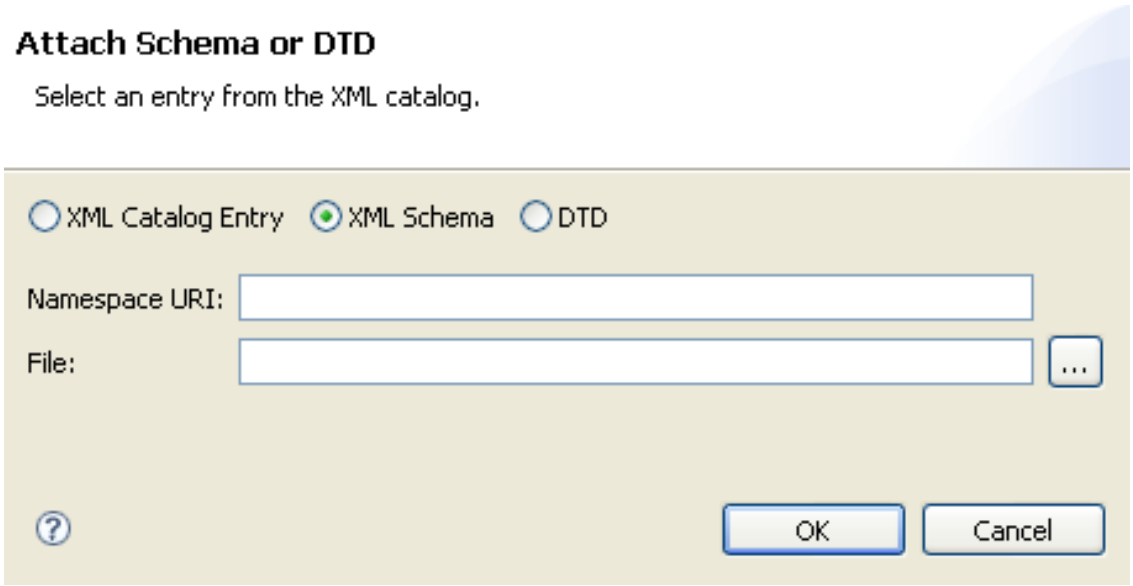
2 Specify the data source (**XML Catalog Entry**, **XML Schema**, or **DTD**) by clicking the appropriate radio button.

3 Provide the necessary information for the selected data source, then click **OK**.

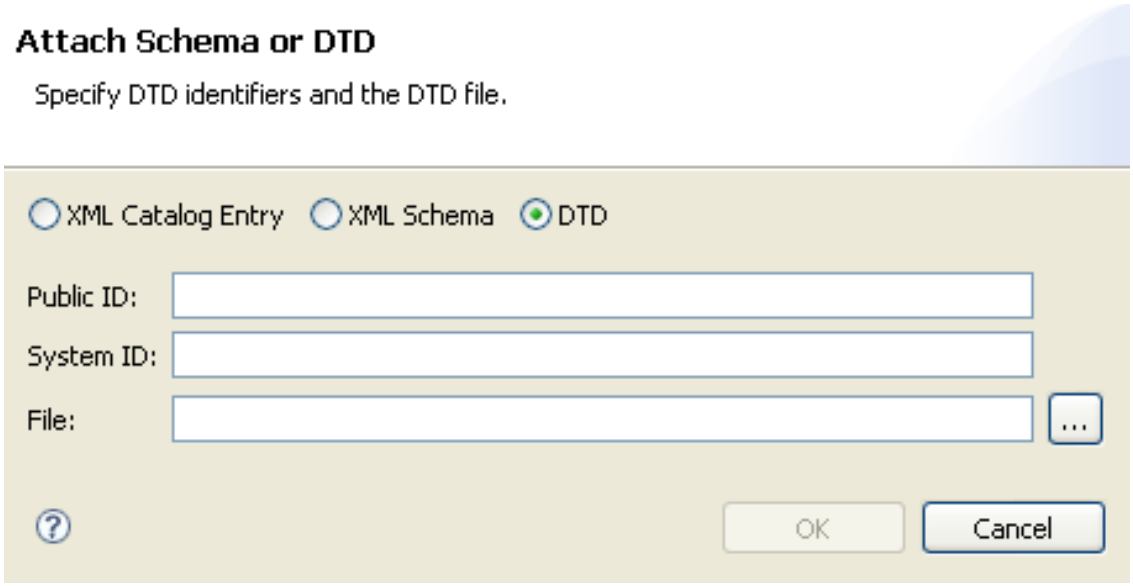
XML Catalog Entry: Choose the appropriate entry from the **XML Catalog Entry** drop-down list.



XML Schema: Specify the namespace URI and the schema file.



DTD: Specify the Public and System IDs and the DTD file.



Setting XML Editor Preferences

You can customize some NetIQ XML editor behaviors by setting preferences. You access the preferences page through **Windows > Preferences > XML**. You can learn more about these preferences in "XML" on page 522.

17 Tools

Designer provides a variety of additional tools to help you manage OpenText Identity Manager projects. This section describes the tasks available through these tools and services:

- "Converting Earlier Projects" on page 417
- "Migrating Driver Configuration Data to a New Server" on page 421
- "Opening a Web Browser" on page 428
- "Launching iManager" on page 428
- "Checking Your Projects" on page 429
- "Configuring TLS for eDir-to-eDir Drivers" on page 435
- "Using DS Trace" on page 438
- "Working with Generic Resources" on page 441
- "Updating Designer" on page 443

For information about managing workspaces, perspectives, and views, see "Workspaces, Perspectives, and Views" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*. For information about editors, builders, and wizards, see "Editors, Builders, and Wizards" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

Converting Earlier Projects

Designer stores projects and configuration information in a workspace. These workspaces are not compatible from one version of Designer to another. Previous Designer workspaces are not compatible with this version of Designer. If you are using an earlier workspace with this version of Designer, you must close all open editors before pointing Designer to an old workspace. Otherwise, the Project Converter does not successfully create a back-up of your project before converting the project to the new file format.

When you start this version of Designer for the first time, you must point it to a new workspace. You can import your existing projects to the new workspace. For more information, see Chapter 11, "Importing into Designer," on page 281.

NOTE: Projects converted to this version of Designer are not backward compatible. In a teaming environment, all the team members must use the same version of Designer when working on a converted project that is stored in a version control system.

- "Converting Earlier Projects" on page 418
- "Converting Projects with the Project Converter Wizard" on page 418

Converting Earlier Projects

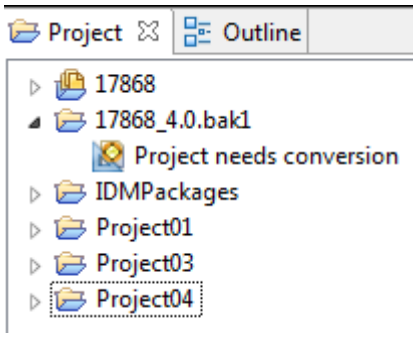
You can import earlier projects from the file system or from the version control system. The conversion supports the objects that are newly added to Designer.

Converting Projects with the Project Converter Wizard

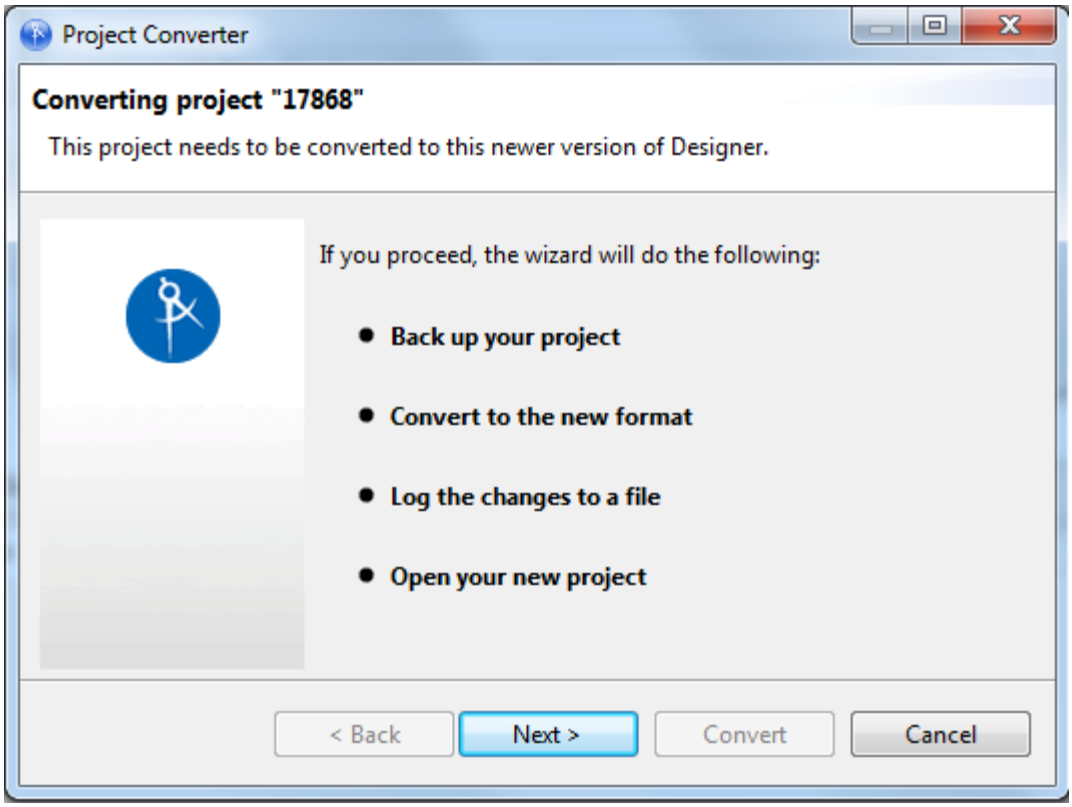
To convert an earlier project:

1. To convert projects that were not open in an editor when Designer was closed, open the project by doing one of the following:
   - Double-click the project in the Project view.
   - Right-click the project in the Project view, then select **Open**.

   Although you can open a project in the Navigator view by clicking the project's `.proj` file, we recommend that you use the Project view instead. Otherwise, the Navigator view takes you into the raw file system.
2. In the Project view, expand the project, then double-click **Project needs conversion**.

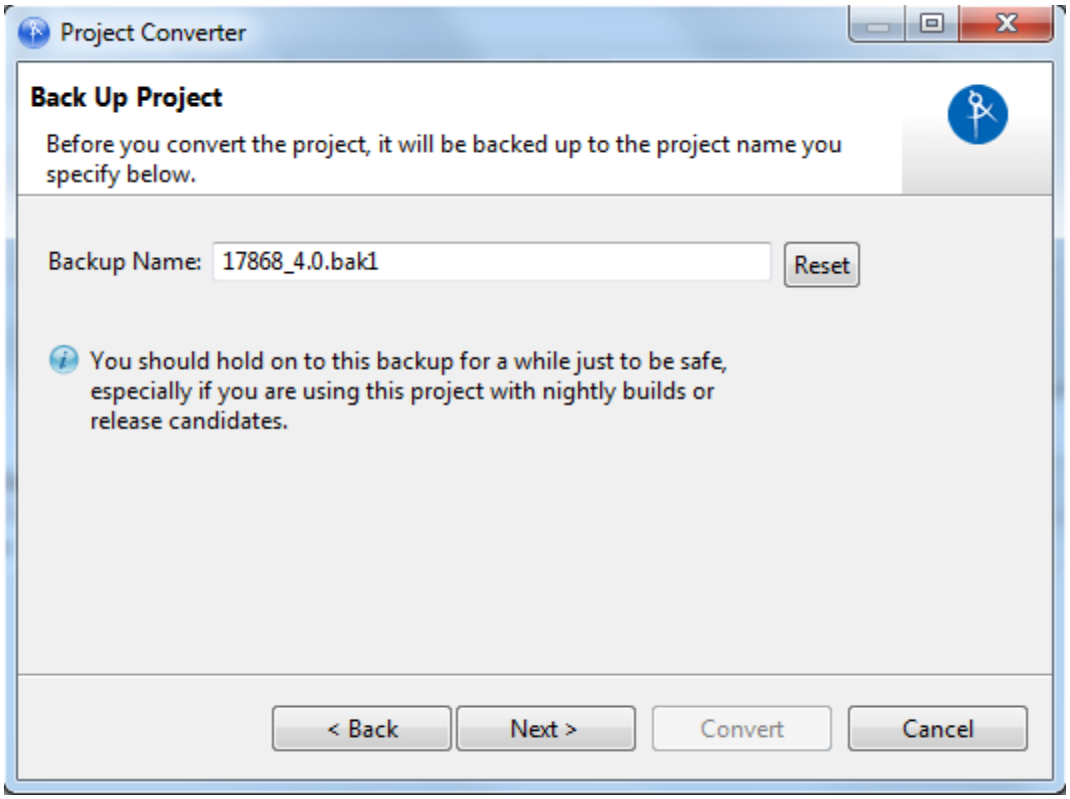


3. Designer opens the project in the Project Converter Wizard. Review the steps, then click **Next**.



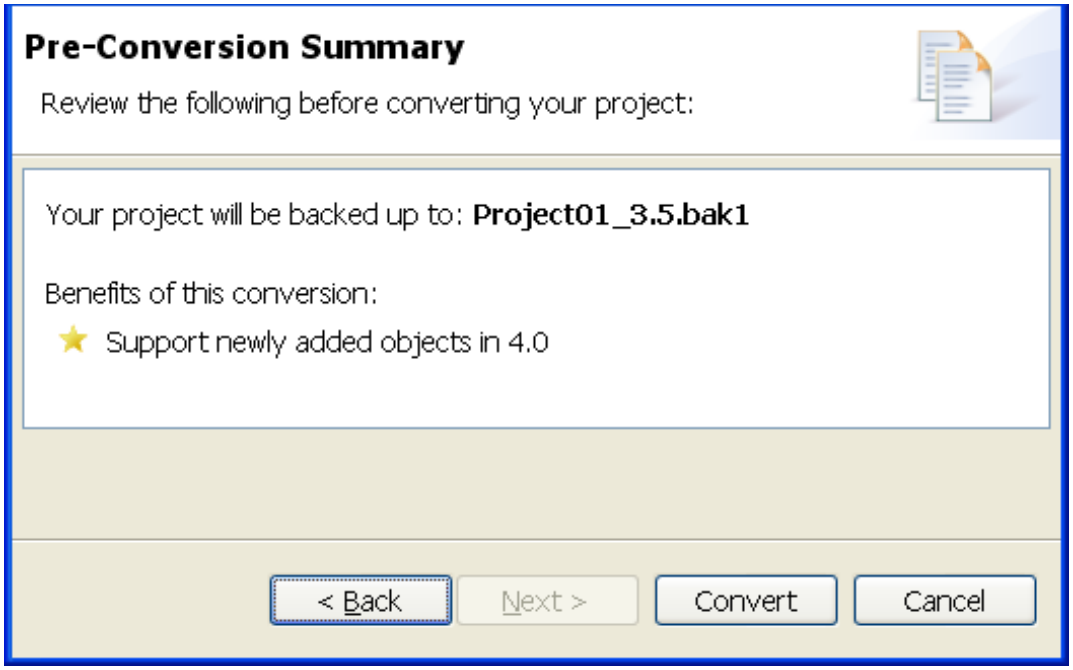
4 Name the project, then click **Next**.

The Project Converter backs up your project before converting. You can accept or change the default name.



5 (Optional) If you edited the name but want to return to the default, click **Reset**.

6 Convert the project by clicking **Convert**.



The converter changes, adds, and removes references, attributes, and elements. It might also create new files or delete old ones. It converts the project file to the new, correct file format. A progress bar displays during the backup and conversion. Converting very large projects might take a few seconds.

7 View the conversion log by clicking **View Log**.

The `conversion.log` file is in the project folder in the `Workspace` directory (for example, `c:\documents and settings\skopai\digitalairlines\conversion.log`).

8 Open the project.

Regardless of the internal format, Designer always deploys to the proper format of the target OpenText Identity Manager environment.

The converter ensures only forward compatibility. It is not backward compatible. A project that is converted to a newer release of Designer cannot be converted to an older release. In order to return to an earlier format, use the backup file of your project.

Migrating Driver Configuration Data to a New Server

If you have added a new server (right-click the Identity Vault and select **New > Server**), you might need to migrate the server data from an existing driver set to the new server. You can do this in one of three ways:

- "Using the Server Migration Wizard to Migrate the Driver Set" on page 421
- "Migrating a Driver Set to a Server in a Different Tree" on page 424
- "Migrating Server Data for Each Driver" on page 427

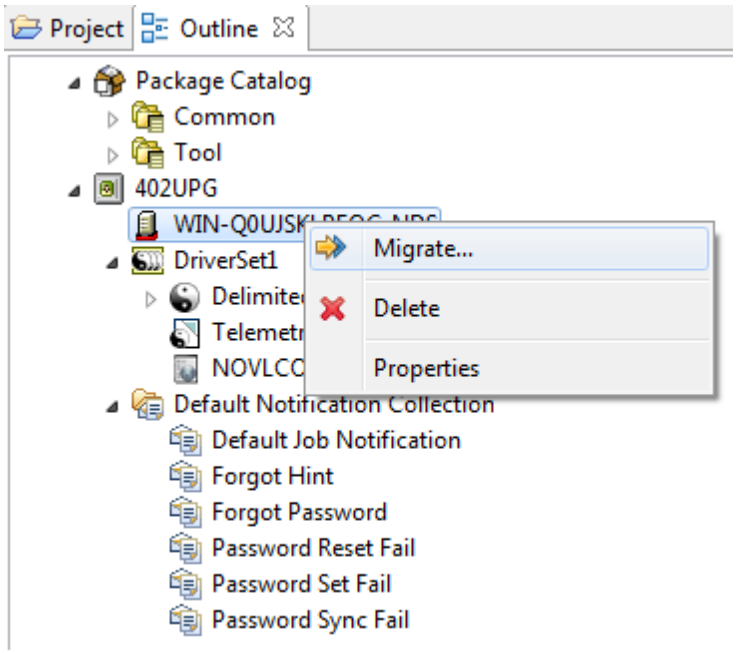
After the server data is migrated, you must redeploy the driver set to the new server in order for the server to become active. For more information, see "Deploying a Project to an Identity Vault" on page 388.

Using the Server Migration Wizard to Migrate the Driver Set

Use the Server Migration Wizard to migrate server-specific data in an existing driver set to a new server. The Server Migration Wizard copies the following server-specific information for the driver set and associated drivers:

- Global configuration values (GCVs)
- Engine control values (ECVs)
- Named passwords
- Driver authentication information
- Driver startup option
- Driver parameters

1 From the Outline view, right-click the server with the associated driver set you want to migrate, then select **Migrate**.



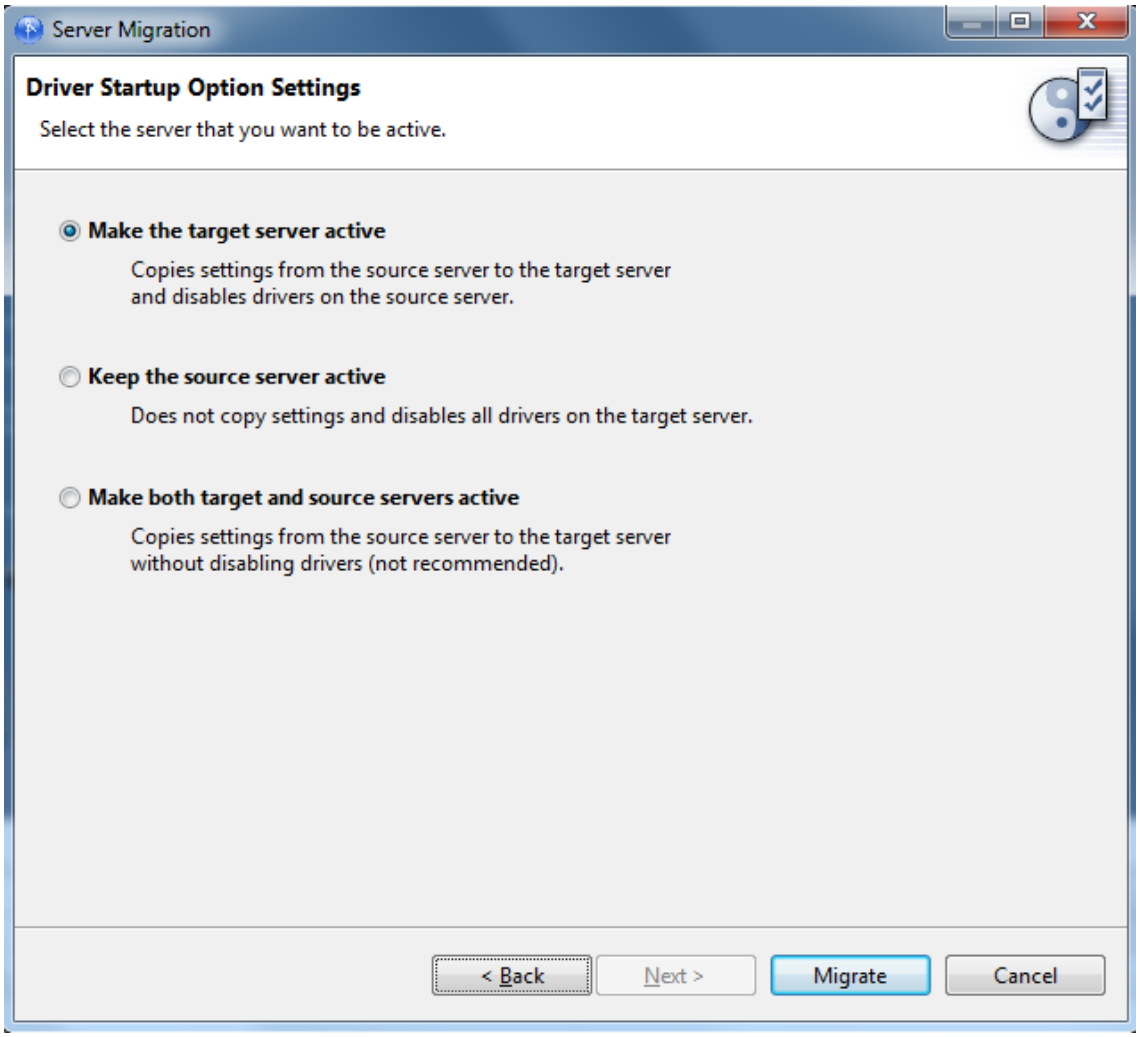
The Server Migration overview page explains that you are migrating a driver set from its source server to a target server along with its server-specific data.

2 Click **Next**.

3 On the Select Target Server page, select the server targeted for driver set migration and select **Next**.

The Target Server list shows only servers that are not presently associated with any driver set and have an OpenText Identity Manager version that is equal to or newer than the source server.

4 In the Driver Startup Option Settings page, select the server that you want to be active.



The default selection is **Make the target server active**. This option copies the current driver startup settings from the source server to the target server and disables all of the drivers on the source server.

The **Keep the source server active** option copies the current driver startup settings from the source server to the target server and then disables the drivers on the target server.

The **Make both target and source servers active** option copies the current driver startup settings from the source server to the target server and does not disable any drivers on either server. This option is not recommended, because having all service queues active on both servers causes the servers to run the same tasks, which can produce unpredictable behavior.

Settings in the Driver Startup Option Settings page only affect the DirXML-DriverStartOption attribute on drivers and not the migration of other server data. You can also set the driver startup options on the driver's **Properties > Driver Configuration > Startup Options** tab. Driver startup options are **Auto Start**, **Manual**, and **Disabled**.

5 Select **Migrate**.

The wizard copies the server-specific information for the driver set and associated drivers to the target server while displaying a progress bar. When the migration finishes, you see The `server has been successfully migrated!`

6 Click **Close** to close the Server Migration Wizard.

7 After the wizard closes, right-click the driver set object in the Outline view and select **Live > Deploy**.

8 If necessary, fill in any needed information in the Identity Vault Credentials window to authenticate to the Identity Vault, then click **OK**.

   You see the Operation In Progress window, followed by the Deployment Summary page, which shows what is being deployed to the Identity Vault.

9 Click **Deploy**.

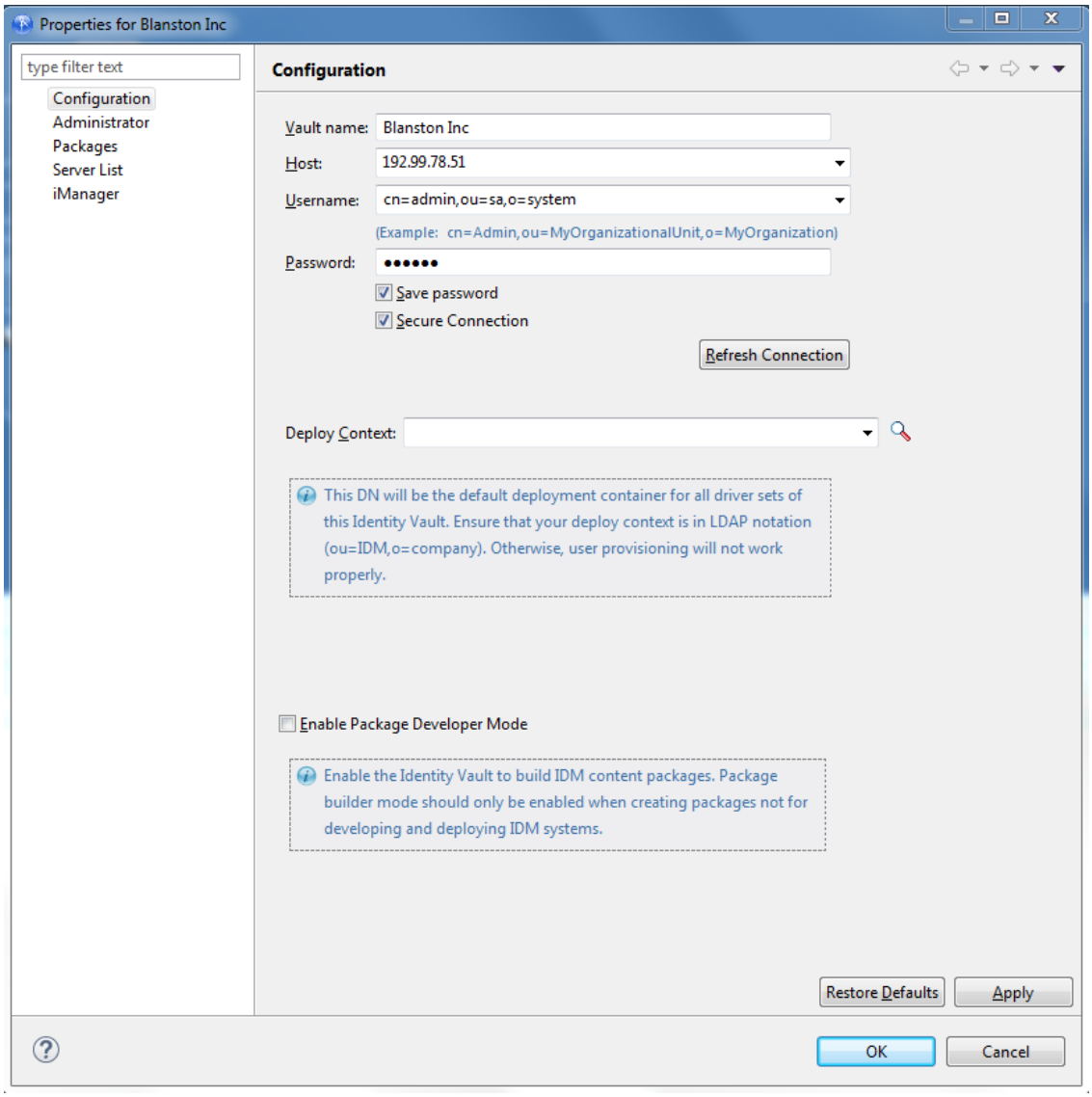
10 If you see errors on the Deployment Results page, click the error to see a summary of the cause and possible solutions. Click **OK** to close.

Migrating a Driver Set to a Server in a Different Tree

This procedure assumes that you have created a new tree and server, but you want to use an existing driver set.

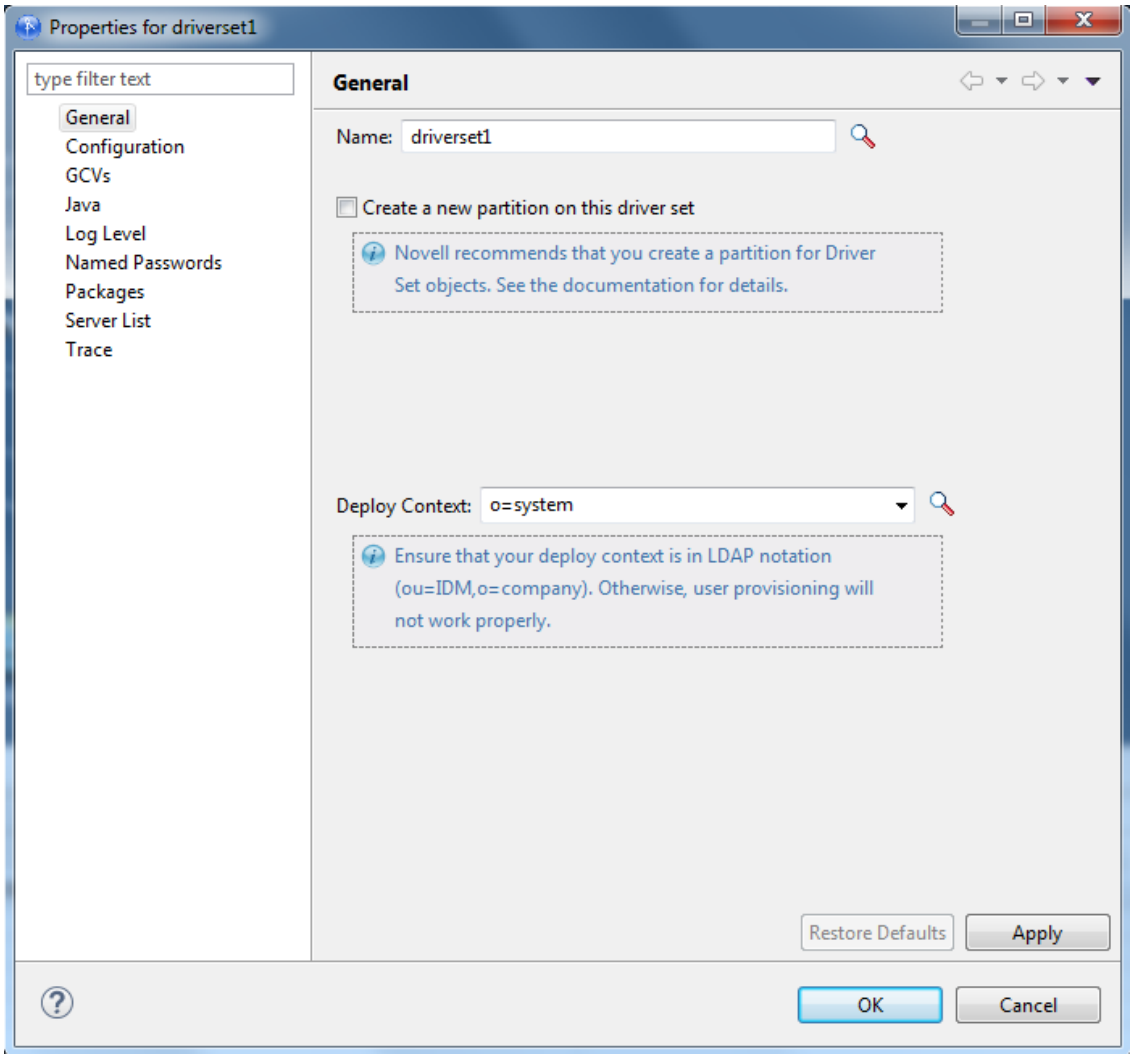
1 Right-click the Identity Vault in the Modeler or Outline view and select **Properties**.

2 In the Configuration section, edit the **Host**, **Username**, and **Password** entries to connect to the new tree, then click **OK**.



3 Right-click the driver set in the Modeler or Outline view and select **Properties**.

4 In the General section, edit the **Deploy Context** to reflect the container where you want to store the driver set. Type the name of the correct container or use the **Browse** icon to find the new container, then click **OK**.



5 Right-click the server object in the Outline view and select **Properties**.

6 Under the **General > Properties** section, edit the **Name** and **Context** entries to match the server in the new tree, then click **OK**.

7 Redeploy the driver set to the new server by right-clicking the driver set object in the Modeler or Outline view and selecting **Live > Deploy**.

You see the Operation In Progress window, followed by the Deployment Summary page, which shows what is being deployed to the new Identity Vault.

8 Click **Deploy**.

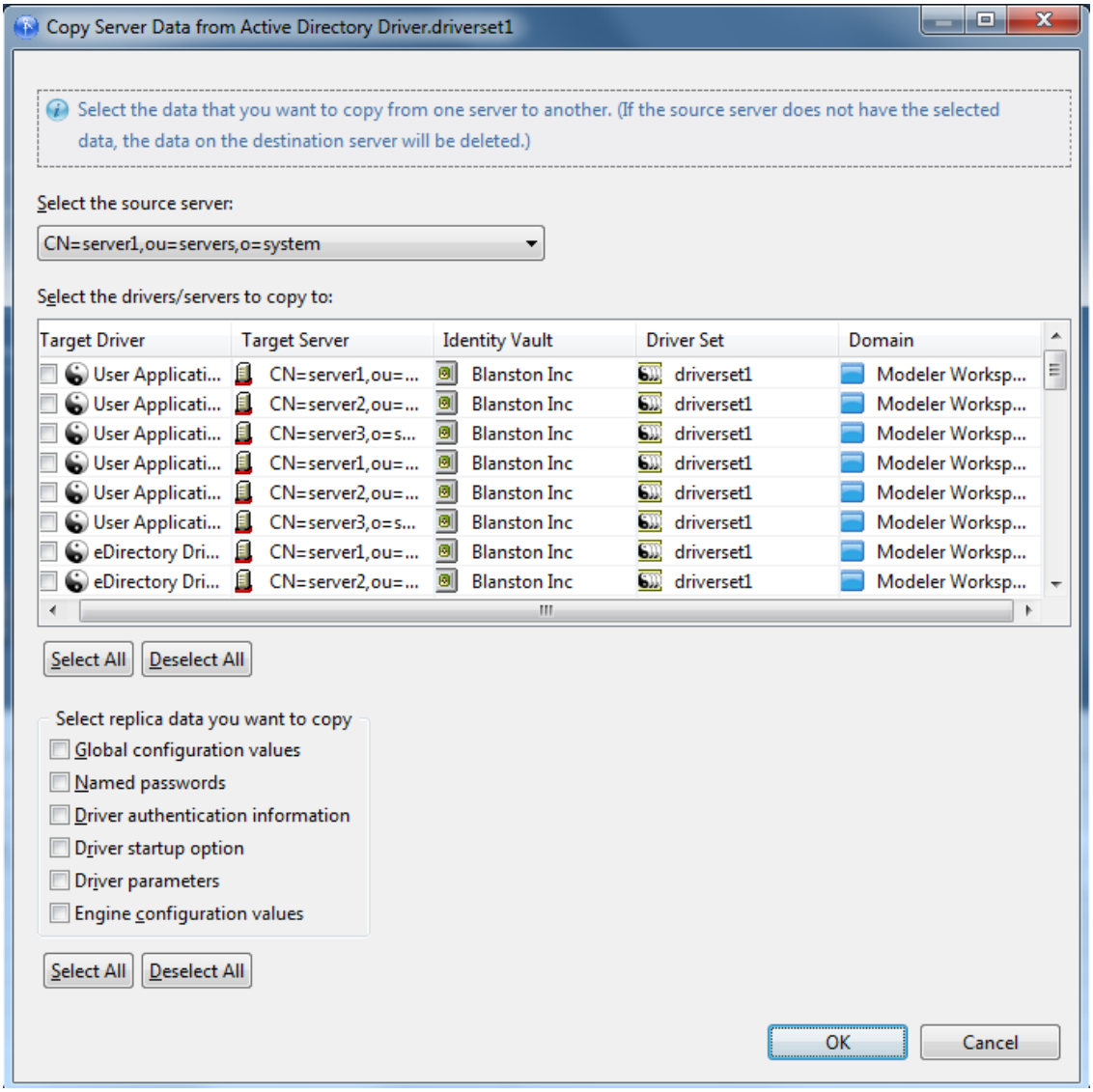
9 If you see errors on the Deployment Results page, click the error to see a summary of the cause and possible solutions. Otherwise, click **OK** to close.

All server-specific data for the driver set is copied to the new server on the new tree.

Migrating Server Data for Each Driver

Although using the Server Migration Wizard is the preferred method, you can also migrate server data for a single driver in the driver set. You can either perform this action for each driver in the driver set, or use the Server Migration Wizard as described in "Using the Server Migration Wizard to Migrate the Driver Set" on page 421.

1 Right-click a driver in the Outline view and select **Copy > Server-Specific Settings**.



2 In the Copy Server Data from Driver.Driver Set window, select the source server. This is the server whose data is copied to the selected targets.

3 Under the **Select the drivers/servers to copy to** entry, select the target driver or drivers on the target server that you want to copy to. This example selects the Active Directory driver as the target driver on the Terabyte5.novell target server.

IMPORTANT: Some server data is specific to a driver type, but other data, like the driver startup option, is not. Know what you want to accomplish before copying one driver's server data to other driver types. Otherwise, drivers on the target server might behave erratically or fail.

4 In the **Select replica data you want to copy** section, select the data you want to copy to the target server. The copied data includes:

- Global configuration values (GCVs)
- Named passwords
- Driver authentication information
- Driver startup option
- Driver parameters

5 After you select the data, click **OK**, then click **OK** in the Complete window.

You must perform this action for each driver in the driver set, or use the Server Migration Wizard.

Opening a Web Browser

You can open a Web browser from within the Designer utility. The Web browser icon is available from the main toolbar.

When you first launch the browser, you are prompted for a home page. After you enter the URL, it is stored in Preferences.

To change the URL:

1 Select **Window** > **Preferences**.

2 Select **Designer for IDM**.

3 Click the **Browser** tab.

4 Type the new URL, then click **OK**.

You can also open an internal Web browser view by selecting **Window > Show View > Other** and then selecting the **Internal Web Browser** option under the **General** heading.

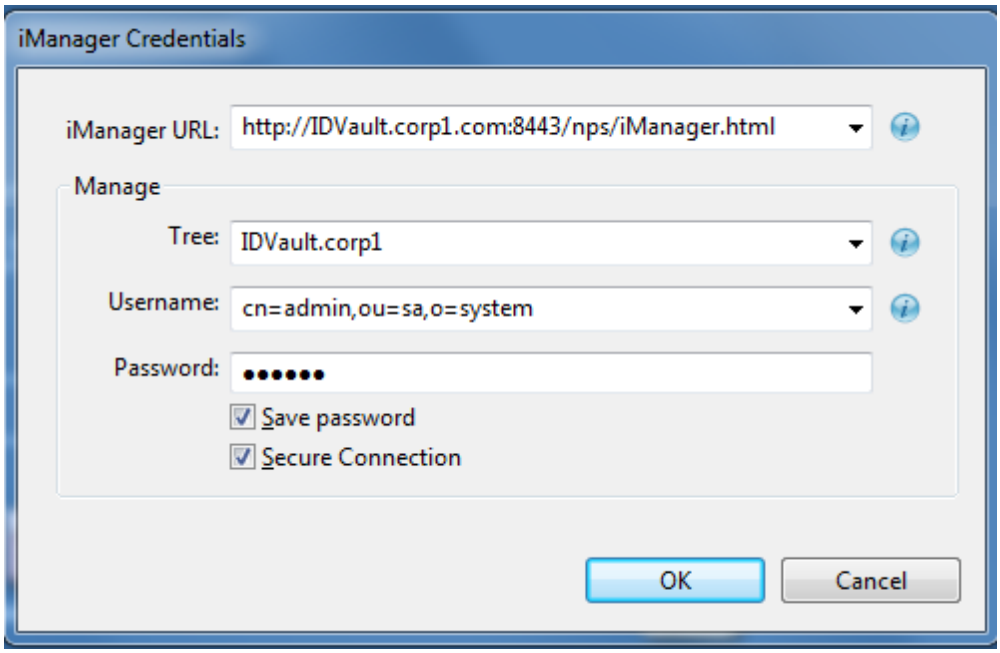
Launching iManager

To launch iManager from within Designer:

1 Right-click the Identity Vault, then select **Live** > **iManager**.

You can also select **Tools** > **iManager**.

2 In the iManager Credentials dialog box, specify the appropriate iManager URL and user credentials to access iManager.



You must specify the iManager URL along with a server name (or IP address) with a replica of the directory tree, username, and password.

Select **Save password** to store the credentials in a history.

3 Click **OK**.

Checking Your Projects

Designer provides the Project Checker so you can check your project. The Project Checker checks for proper design, contexts, server associations, policies, missing user data, and dependency problems that can cause a project deployment into the Identity Vault to fail. You can check a project at any time, but you should definitely run the Project Checker before deploying a project.

- "Checking a Project" on page 429
- "Customizing the Project Checker" on page 431
- "Items That Are Checked" on page 433

NOTE: Project Checker only checks the objects in Designer. It does not check the current objects in the Identity Vault.

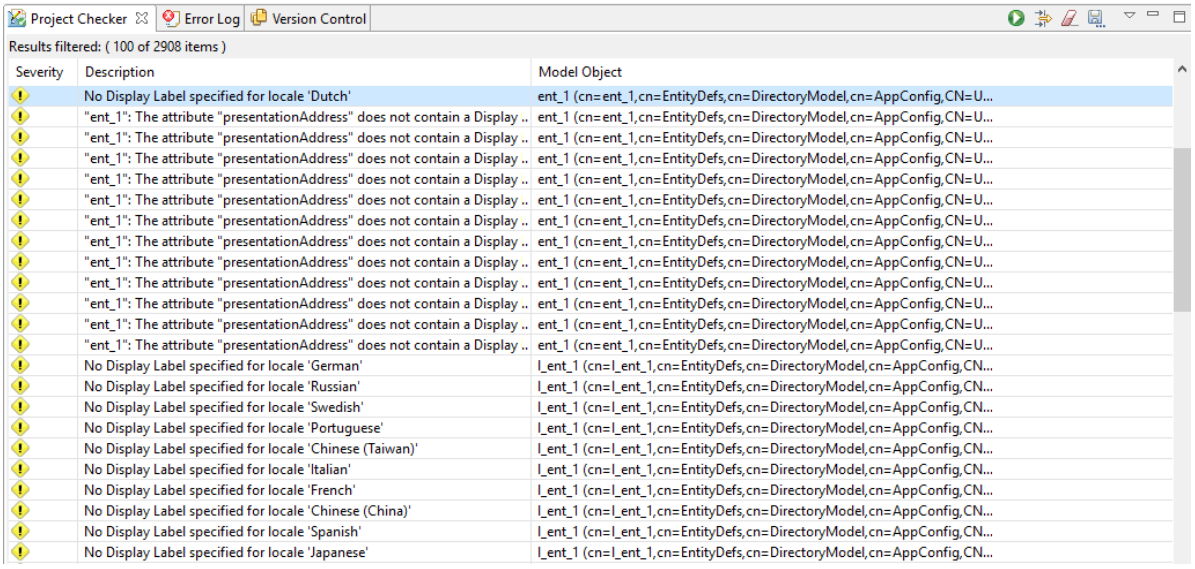
Checking a Project

1 In the Project or Outline view, select the project, then select the **Launch Project Checker** icon in the Designer toolbar.

The Project Checker is also available from the **Window** > **Show View** menu.

2 Click the **Run the Project Checker** icon .

If you have not saved the project, Designer prompts you to save it.



The Project Checker displays a list of versioning conflicts, errors, warnings, and information messages about the project. In the Project Checker view, you can do the following:

Action	Description
See detailed information about a list item	Double-click a list entry to open a properties page that displays the following information about the entry: ◆ The message severity ◆ A message description ◆ The model object that caused the message ◆ The line number where the problem occurred, if available ◆ Details about the message, if available ◆ A recommended solution for the message, if available
Sort the list	Click any header in the Project Checker to sort the entry list on that parameter (Severity, Description, and Model Object). By default, Project Checker sorts entries by severity in descending order (most current at the top of the list.)
Filter the list	Click the Configure Filters icon to customize the Project Checker. For more information, see "Customizing the Project Checker" on page 431.
Clear the list	Click the Clear Results icon to clear the Project Checker entry list.
Save the list	Click the Save Project Checker Results to a File icon to save the current Project Checker entry list to a text file so you can review it off-line.

Action	Description
Menu options	Click the Menu icon to select one of the following: ◆ View the messages in a hierarchical layout, according to functions (OpenText Identity Manager, provisioning, etc.) ◆ View the messages in a flat layout (default). ◆ Automatically check the project when you save it. ◆ Configure filters ◆ View the Project Checker's Preferences page.

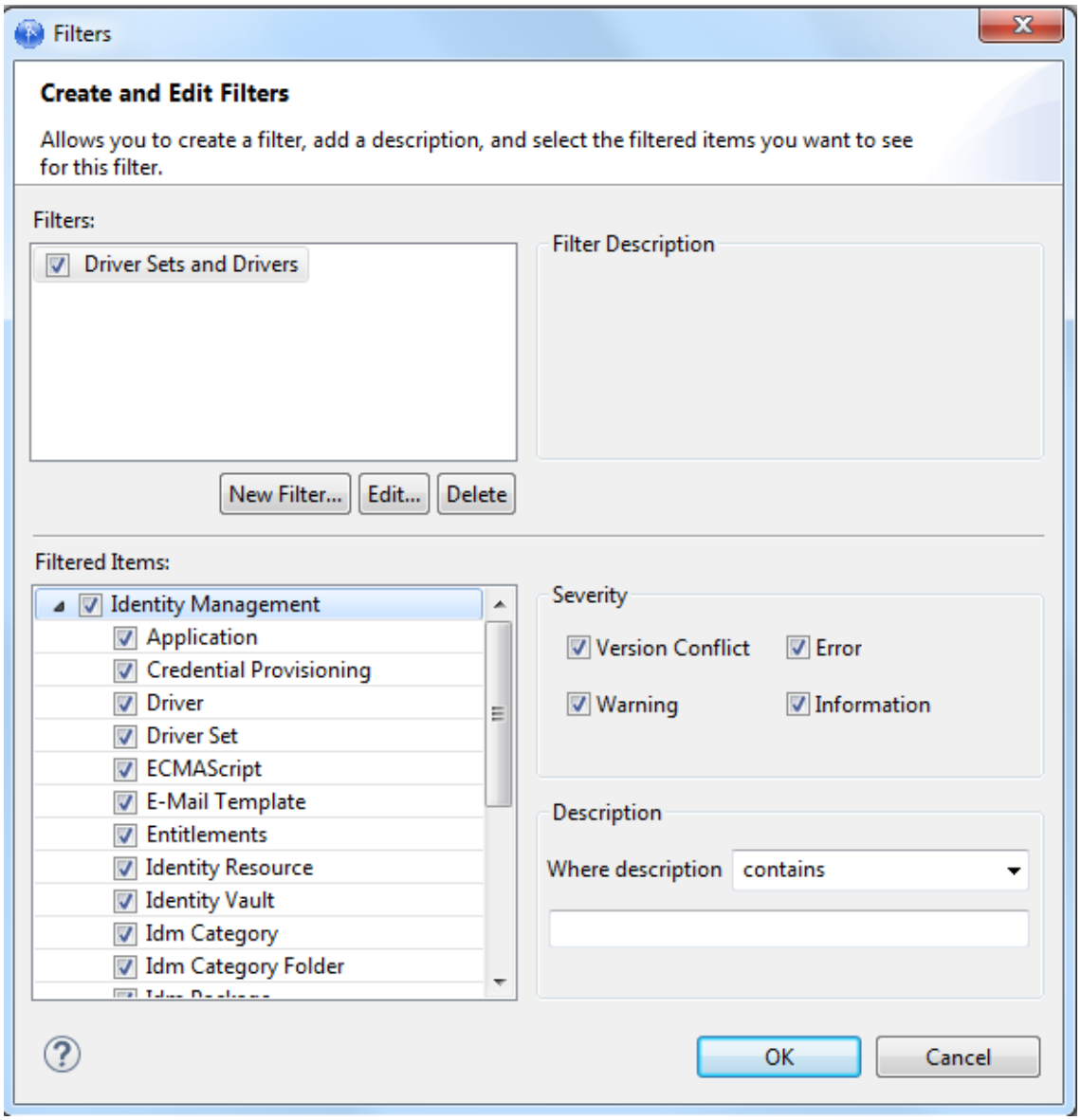
Customizing the Project Checker

You can customize the Project Checker by creating and editing filters. The filters allow you to receive messages about the items you want to verify. You can create multiple filters, but only one filter can be used at a time.

To create a filter:

1. In the Project Checker, click the **Configure Filters** icon .
2. Click **New Filter**.
3. Specify a name and description for the filter.

   You can select which items are checked, what types of messages are returned about the items, and use key words to limit the messages returned. For example, you can search for all messages about the Driver Set and Driver objects that contain the word "attribute."



4 Click **OK**.

To edit the name and description of the filter:

1 Select the filter, then click **Edit**.

2 After you have completed the changes, click **OK**.

To delete a filter:

1 Select the filter.

2 Click **Delete**.

Items That Are Checked

The Project Checker looks at specific items in the project. It checks the items in the User Application as well as the rest of OpenText Identity Manager.

The following table describes the specific items that are checked. The list increases with each release of Designer.

Table 17-1 *OpenText Identity Manager Items That Are Checked*

Item	Description
Driver	◆ Checks for the presence of a Schema Mapping policy. ◆ Checks for an invalid Active Directory container. ◆ Checks the trace level setting. If it is set to more than 0, an informational message is displayed. ◆ Checks to see if the LoopBack driver is being used instead of the OpenText eDirectory driver. ◆ Verifies that the GUID attribute is set to synchronize on the Subscriber channel. ◆ Verifies that the GUID attribute is not set to synchronize on the Publisher channel. ◆ Checks the classes on the Publisher and Subscriber channels that are set to **Ignore** and verifies that the attributes for these classes are not set to **Synchronize**. ◆ Checks for the presence of a filter and makes sure it is not empty. ◆ Checks to make sure that the Publisher Placement policy does not contain **set operation destination DN** or **set xml attribute** operations. ◆ Checks for the presence of a Publisher Placement policy. ◆ Checks to make sure that no policy on the Publisher channel contains **set operation destination DN** or **set xml attribute** operations. ◆ Checks to make sure that the Subscriber Placement policy does not contain **set operation destination DN** or **set xml attribute** operations. ◆ Checks to see if the Subscriber Placement policy is missing. ◆ Checks to make sure that no policy on the Subscriber channel contains **set operation destination DN** or **set xml attribute** operations. ◆ Checks to make sure that the npsmDistributionPassword attribute and the public-private key pair attributes do not simultaneously exist in the User class. ◆ Checks to make sure that the authentication method on the Active Directory driver is set to **Negotiate** when synchronizing passwords. ◆ Checks the filter for invalid data. ◆ Checks the driver to see if it is publishing both NDS and Distribution passwords. If it is, this is an invalid setting. ◆ Checks for the presence of the nspmDistributionPassword attribute in the User class in the Filter, if password synchronization is enabled. ◆ Checks that the nspmDistributionPassword attribute is set to sync or notify, if password synchronization is enabled.
Driver Set	◆ Checks to make sure that the deployment context for the Driver Set object is set. ◆ Checks to make sure that a server object is associated with the Driver Set object.

Item	Description
E-mail Template	Checks to see if the e-mail notification template is empty.
Entitlements	◆ Checks to see if the driver supports entitlements. ◆ Checks to see if the attribute DirXML-EntitlementRef is added to the Subscriber channel, if there are policies that use entitlements in the driver. The DirXML-EntitlementRef must be set to Notify or Synchronize for the entitlements to work.
ECMAScript	Checks to see that the ECMAScript object can run.
Identity Vault	◆ Checks to see if the username to authenticate to the Identity Vault is missing. ◆ Checks to see if the hostname for the Identity Vault server is missing. ◆ Checks to see that the password for the user is not stored in the project.
Job	Checks to see that the job object can run.
Library	Checks to see that the library object can run.
Mapping Table	◆ Checks to see that the mapping table object can run. ◆ Checks to see if there is an empty column name. ◆ Checks to see if there is a duplicate column name.
Policy	◆ If there are global configuration values in the policy, it checks to make sure they exist on the Driver or Driver Set object. ◆ Checks to see if local variables are defined before they are used. ◆ Validates the policy against the DTD.
Schema	◆ Checks to see if the class is missing from the schema. ◆ Checks to see if attributes are missing from the schema. ◆ Checks to see if the attribute for the class is missing from the schema.

Table 17-2 *Provisioning Items That Are Checked*

Item	Description
Configuration	Verifies that the XML is well-formed and complies with the schema that defines the elements needed for entities, attributes, lists, relationships, and so on.
Entity	◆ Checks every entity to ensure that references to other entities and global lists are valid. ◆ Ensures that every entity has at least one attribute defined.
List	Ensures that every local and global list contains at least one item.
Org Chart Relationship	Verifies that the entities and attributes of a relationship have been deployed.
Provisioning Request Definition	Verifies that a workflow follows rules for activities and flow paths.

Configuring TLS for eDir-to-eDir Drivers

If you want the eDir-to-eDir drivers to communicate securely, you must perform the following tasks:

- "Prerequisites" on page 435
- "Enabling TLS" on page 435
- "Creating Certificates" on page 437

Prerequisites

- Identity Vaults exist in your physical network tree as well as in the Modeler.
- Each Identity Vault is set up. Otherwise, you are prompted for setup information when you try to create certificates.
- Each driver set is associated with a server.
- Using the eDir-to-eDir driver's General property page, verify that each driver has a name and a deploy context. The context might be inherited from the driver set.
- The eDir-to-eDir drivers have been deployed. Otherwise, certificates cannot be created.

  To find out whether the driver has been deployed:

  1. Right-click the eDir-to-eDir driver.
  2. Click **Live** > **Deploy**.
  3. In the eDir-eDir Driver Deployment dialog box, click **No**.

  If the driver has been deployed, the **Compare Status** field in the Deployment Summary dialog box displays **Equal** or **Unequal**. Otherwise, the field displays **Not Deployed**.

  After objects have been deployed, the objects should show as equal unless passwords are set in OpenText eDirectory that are not set in Designer. Designer does not deploy passwords unless they are specifically set in Designer. This exception prevents overwriting passwords in OpenText eDirectory because Designer cannot import them.

Enabling TLS

1 Launch the TLS Configuration dialog box.

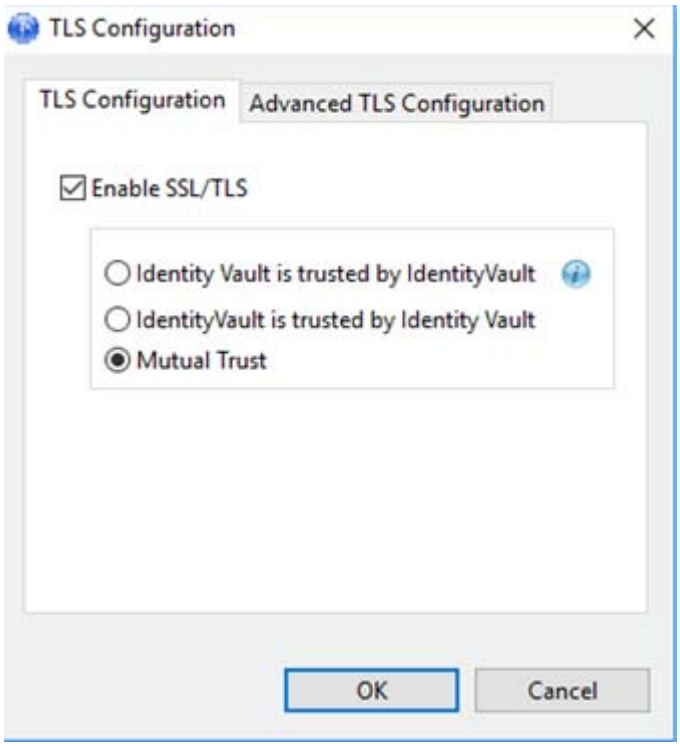
  A common way to launch the dialog box is to right-click the eDir-to-eDir application, then click **Secure Connection Settings**.

  Other launch points:

  - Select the eDir-to-eDir application, then click **Model** > **eDir-to-eDir** > **Secure Connection Settings**.
  - Right-click **eDir-to-eDir** in the Outline view, then click **Secure Connection Settings**.
  - Right-click an eDir-to-eDir driver, click **Properties** > **Driver Configuration** > **Authentication**, then click **Configure TLS**.

    The **Configure TLS** icon displays only on eDir-to-eDir driver pages.

2 Click **Enable SSL/TLS**.



3 (Optional) Use the **Advanced TLS Configuration** to select key size, hash algorithm, and validity period.

The validity period is important for when a certificate has expired and you need to overwrite or create a new one.

4 Select a direction of trust.

These options apply to certificates that OpenText creates for OpenText eDirectory. The options do not apply to third-party security certificates.

The default is Mutual Trust, which is considered to be the most secure.

Unless you want to use the certificate for authentication, the option that you select doesn't matter. If only encryption is important, you can select any one of the three options.

If authentication is important, select the option that gives you the appropriate trust.

Scenario: JJ Infrastructure Tree Trusts JT ID Vault. JJ Infrastructure Tree is the organizational certificate authority. JJ Infrastructure Tree signed a certificate and placed it in JT IDVault. JT ID Vault trusts JJ Infrastructure Tree. The two vaults synchronize data through a secure connection.

If the two vaults break their trusted relationship, JJ Infrastructure Tree can prevent sensitive data from being synchronized by revoking its certificate.

Scenario: JT ID Vault Trusts JJ Infrastructure Tree. JJ Infrastructure Tree creates two certificates. One is placed in JJ Infrastructure Tree, and the other is placed in JT ID Vault. The two vaults synchronize data through a secure connection.

If the two vaults break their trusted relationship, JJ Infrastructure Tree can prevent sensitive data from being synchronized by revoking its certificate.

Scenario: Mutual Trust. JT ID Vault and JJ Infrastructure Tree both sign certificates.

5 Click **OK**.

After you click **OK**, Designer does the following:

- Modifies both OpenText eDirectory drivers.
- Locks the **User ID** field, which displays on the driver configuration's Authentication page, because both drivers must use that field.

You can enable or configure TLS without immediately deploying the drivers. You can turn the settings on. However, you can't create SSL/TLS certificates unless the drivers have been deployed into their respective Identity Vaults. If you enable SSL/TLS but want to create certificates later, you can do so.

Creating Certificates

A driver's Properties page enables you to configure a driver so that you can deploy it. Similarly, the **Enable SSL/TLS** option enables you to set up your configuration for TLS, then create and deploy the certificates when you are ready. When you deploy a configured driver set or select **Create eDir-to-eDir Certificates,** Designer creates the certificates in the directory.

This section assumes that you have enabled and configured SSL/TLS for the deployed eDir-to-eDir drivers.

1. In the Modeler, right-click the eDir2eDir application.
2. Click **Live** > **Create eDir-to-eDir Certificates**.

You can also do one of the following:

- Right-click the eDir2eDir object in the Outline view, then click **Create eDir-to-eDir Certificates**.
- The first time that you enable and configure SSL/TLS on driver's **Authentication** tab, click **OK**, then follow the prompts. A Create Certificates dialog box appears. Click **Yes**.

Scenario: Enabling TLS. TLS has not been enabled. Select **Live** > **Create eDir-to-eDir Certificates**. Designer prompts you to enable SSL/TLS. Click **OK**, enable TLS,yes select a direction of trust, and click **OK**. Designer creates certificates.

Scenario: Deploying eDir-to-eDir Drivers. eDir-to-eDir drivers and the driver set are configured. A context displays in the driver set's **Deploy Context** field. You can now deploy the driver set.

To do so, right-click the driver set, click **Live** > **Deploy Driver Set**. Designer prompts you to deploy both OpenText eDirectory drivers. (Otherwise, Designer can't successfully create certificates.) Click **Yes**. Designer builds a deployment summary, then lists items that are associated with the Identity Vaults and will be deployed. To deploy the drivers, click **Deploy**.

Because the driver set is already configured, Designer creates the certificates.

For additional information about eDir-to-eDir certificates, see eDir-to-eDir SSL/TLS in **Preferences**.

Using DS Trace

Designer provides DS Trace so you can monitor DirXML events in your OpenText Identity Manager environment. DirXML events constitute those events accessible by using the **DirXML** and **DirXML Drivers** switches in OpenText eDirectory's DS Trace service.

Designer uses LDAP to obtain this information from the Identity Vault. By default, it uses the default LDAP ports (389 or 636) to establish a connection. If your LDAP service runs on non-standard ports, make sure you specify the correct ports.

DS Trace lets you view both live DS Trace logs, and create and view stored DS Trace log files.

- "Viewing DS Trace Live" on page 438
- "Creating a DS Trace Log File" on page 440
- "Viewing a DS Trace Log File" on page 440

NOTE: The DS Trace view is not the same as the Trace view, which provides information about Designer functionality. For information about the Trace view, see "Trace" on page 494.

DS Trace includes the following icons:

Icon	Description
	The **Resume Trace** icon restarts a live DS Trace session that you have previously stopped. It is not available for DS Trace log files.
	The **Stop Trace** icon stops a live DS Trace session. It is not available for DS Trace log files.
	The **Connect to Server** icon launches the Login Credentials dialog box so you can authenticate to the server where you want to run DS Trace.
	The **Load Trace Log File** icon lets opens a previously saved DS Trace log file.
	The **Save Trace** icon save the current live DS Trace session to a log file.
	The **Search** icon opens a Find/Replace dialog box where you can search the current DS Trace log file for a specific string. It is not available for live DS Trace.
	The **Configure Trace** icon provides access to live DS Trace settings. It is not available for DS Trace log files.
	The **Clear Trace** icon clears all DS Trace entries from the live DS Trace log.

Viewing DS Trace Live

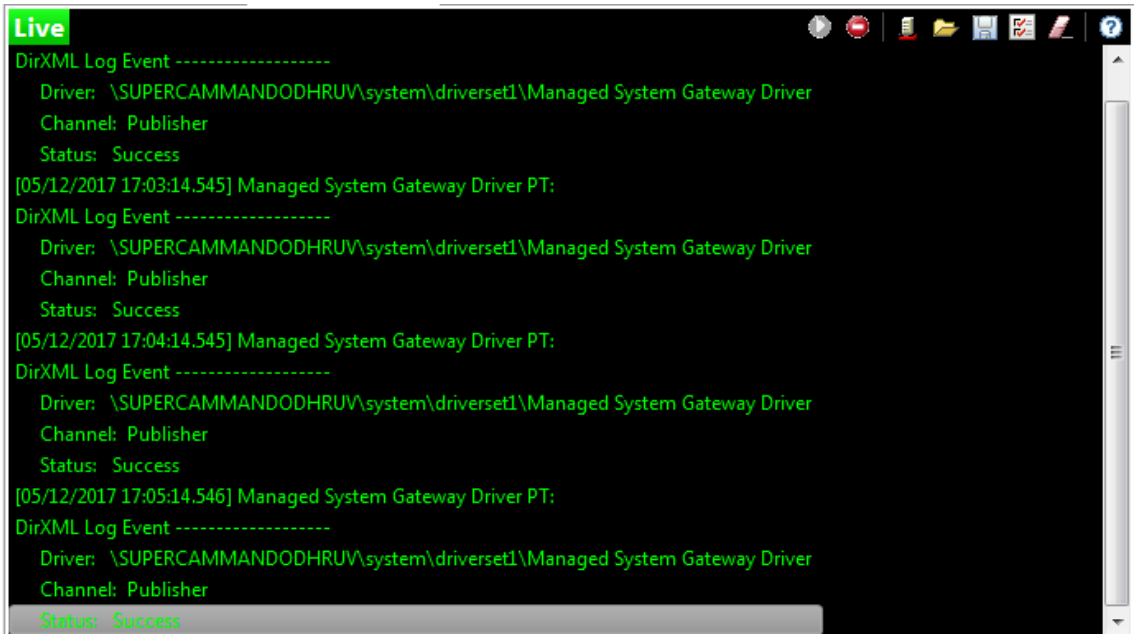
You can view a live DS Trace for any Identity Vault in your OpenText Identity Manager environment.

NOTE: Designer provides live DS Trace preferences that let you specify how many entries to keep in the log and whether or not to auto-scroll the log so you can always see the most current entries. You can edit these preferences in **Windows** > **Preferences**, then select **NetIQ** > **Designer** > **DS Trace** from the left navigation.

If the Identity Vault is in your current Designer project:

1 In the Object view or the Modeler, select an Identity Vault object, then select **Live** > **DS Trace**.

Alternatively, you can right-click the Identity Vault object, then select **Live** > **DS Trace**.



2 Review the live DS Trace session as needed.

By default, the DS Trace session is running. You can stop, resume, clear, and save the current trace to a file by using the icons in the DS Trace view toolbar.

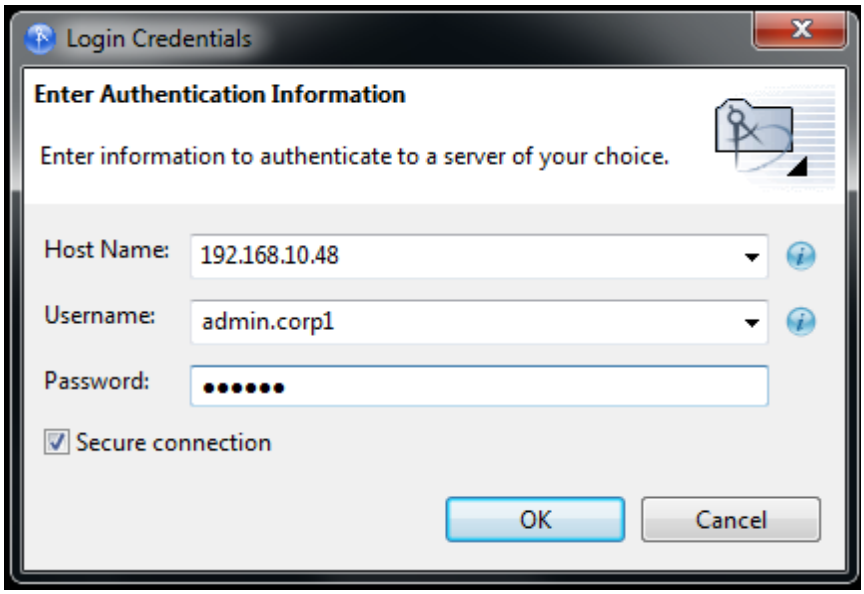
If the Identity Vault is not in your current Designer project:

1 From the main Designer toolbar, select **Tools** > **DS Trace**.

2 In the DS Trace view, click the **Connect to Server** icon.

3 In the Login Credentials dialog box, specify the directory host name (or IP address), username, and password necessary to connect to the appropriate Identity Vault, then click **OK**.

Select **Secure Connection** if you need to use SSL to connect to the Identity Vault server.



You can open a DS Trace session to a different Identity Vault server at any time by clicking **Connect to Server** and providing the appropriate authentication credentials.

4 Review the live DS Trace session as needed.

By default, the DS Trace session is running. You can stop, resume, and save the current trace to a file by using the icons in the DS Trace view toolbar.

Creating a DS Trace Log File

DS Trace lets you create log files of DS Trace entries so you can review them offline.

1 From the live DS Trace view, select the **Save Trace** icon .

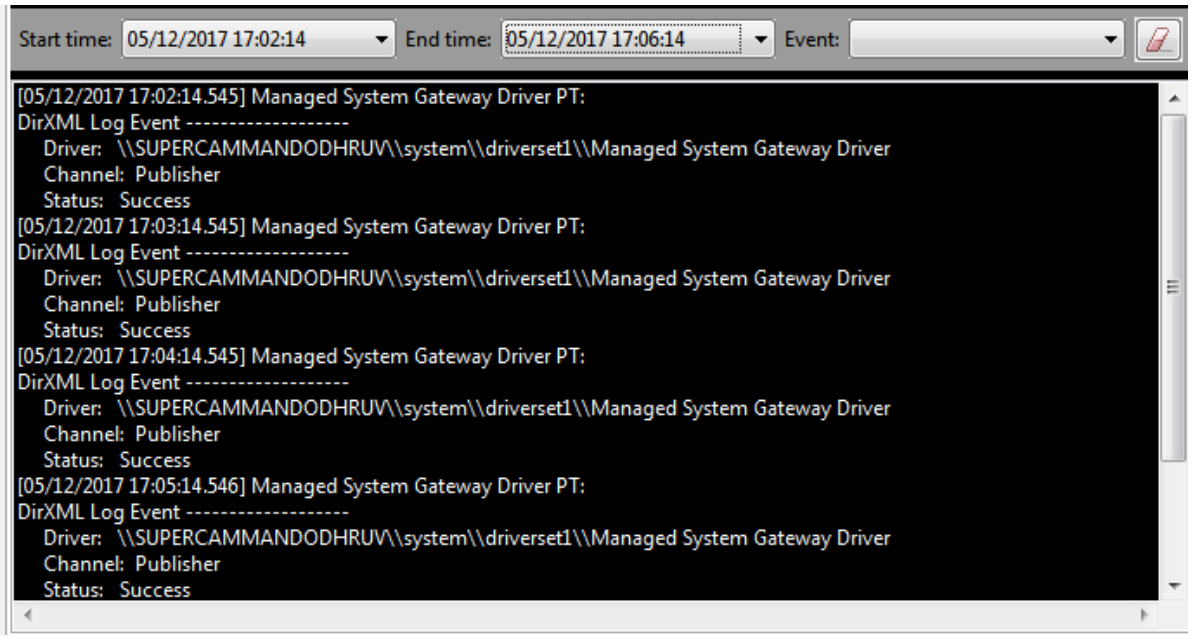
2 Specify a name and location for the log file, then click **Save**.

DS Trace saves the log file as a rich text file (`.rtf`) so it can maintain the color coding used in the live DS Trace view. You can view the log file with any editor that supports the `.rtf` file format.

Viewing a DS Trace Log File

The DS Trace view is an editor that enables you to view DS Trace log files.

Figure 17-1 *The DS Trace View*



To view DS Trace log files:

1. Click **Tools** > **DS Trace**.
2. Select the **Load Trace Log File** icon, then browse to and select the DS Trace log you want to open.
3. Review the DS Trace log file as needed.
   - Use the **Start Time**, **End Time**, and **Event** drop-down lists to filter the trace file. This helps you narrow the displayed trace file data so you can more easily locate specific information.
   - To clear an existing filter, click the **Clear Filter** icon .
   - Select the **Search** icon (in the DS Trace icon bar) to open a Find/Replace dialog box that lets you search for a specific string in the DS Trace log file.

NOTE: The Eclipse text editor does not support color, so when you view a DS Trace file in Designer it displays in black and white. However, because Designer saves the DS Trace log file in standard Rich Text Format (`.rtf`), any external text editor that supports color displays the log file in color, as seen in the live DS Trace view.

Working with Generic Resources

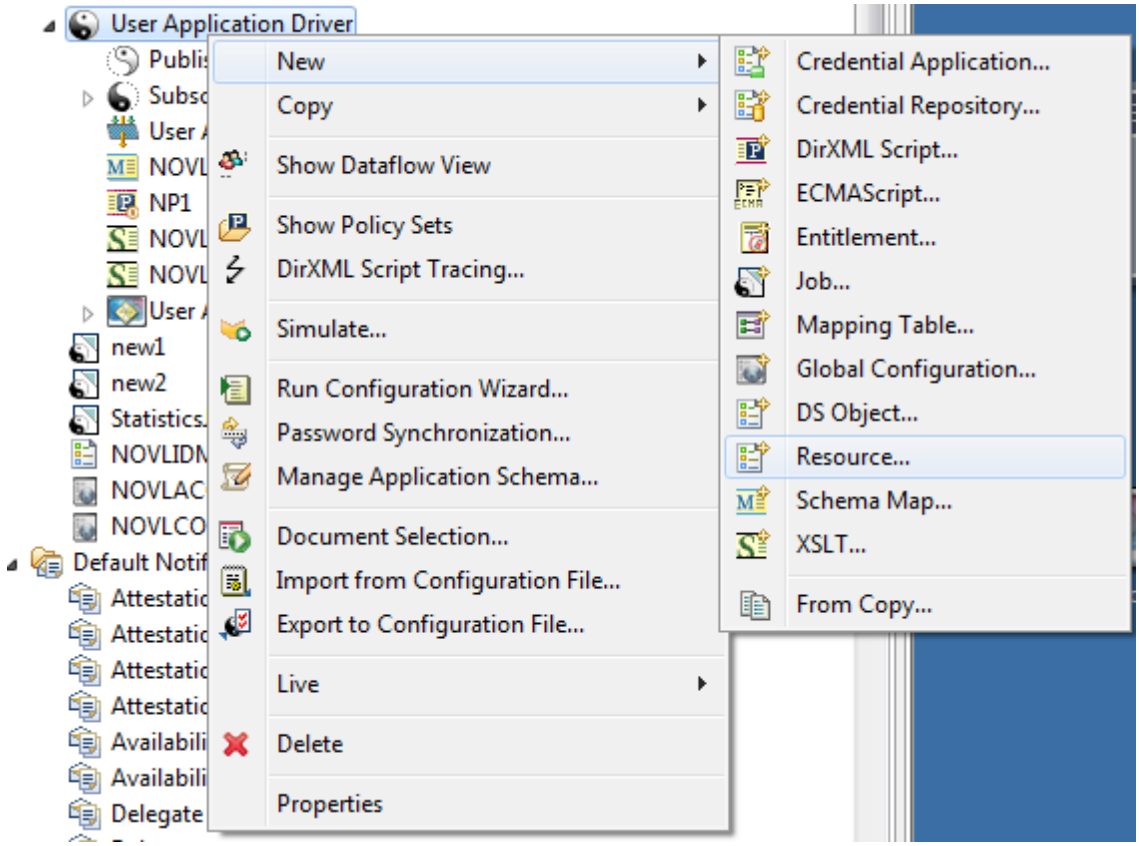
A Resource object is stored in a Driver object or a library. A Resource object stores parameters, which drivers use at any time. When multiple drivers need the same set of constant parameters, the drivers use a Resource object.

A Generic Resource object in Designer enables you to store information in XML or text format. The information can be a piece of documentation, notes, or some piece of data that policies access.

- "Creating a Generic Resource Object" on page 442
- "Editing a Generic Resource Object" on page 443

Creating a Generic Resource Object

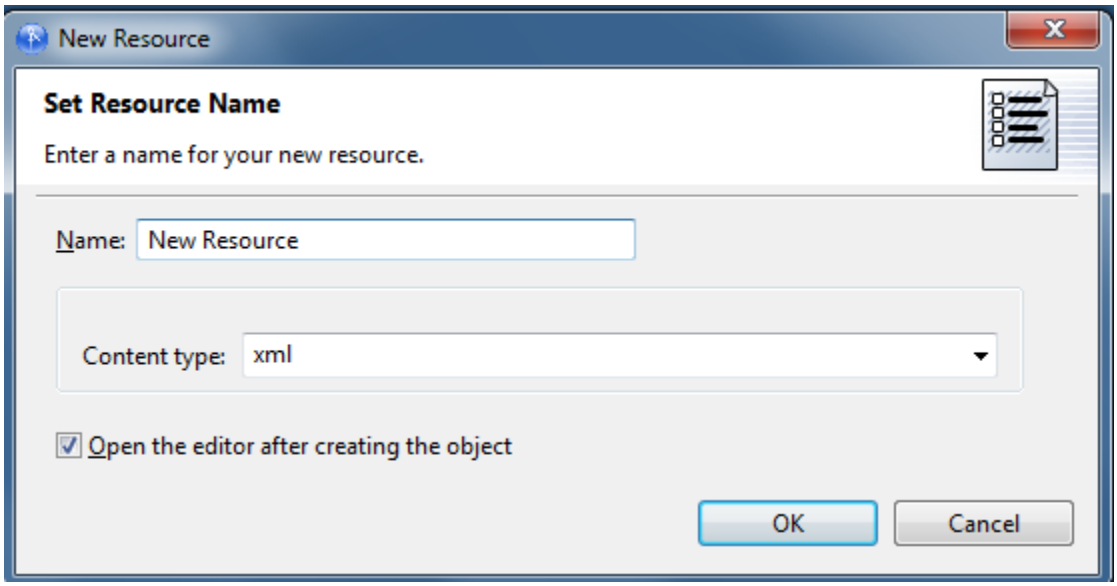
1 In the Outline view, right-click a driver, then select **New** > **Resource**.



You can also do one of the following:

- Right-click a driver, then select **New** > **Resource**.
- With the Dataflow view active, right-click a Subscriber or Publisher channel, then select **New** > **Resource**.
- In the Outline view, right-click a library, then select **New** > **Resource**.

2 Specify the name of the Generic Resource object.



3 Select **XML** or **Text** as the content type.

4 Select **Open the editor after creating the object**, then click **OK**.

5 In the File Conflict dialog box, click **Yes**.

6 Specify the desired XML or text, then press Ctrl+S to save the resource object.

Editing a Generic Resource Object

1 In the Outline view, below the library, right-click the Generic Resource object, then select **Edit**.

2 In the File Conflict dialog box, click **Yes**.

3 Make changes, then save (Ctrl+S).

Updating Designer

When you start Designer, you are prompted about how you want to receive updates. You can change this setting in Preferences.

If you select to not automatically update Designer, you can get updates by using the Help menu or the Welcome page.

To update from the Help menu:

1 Click **Help** > **Check for Designer Updates**.

- If your version of Designer is up-to-date, a prompt informs you that no updates are available.
- If an update is available, a prompt lists components that you can update.
- If your version of Eclipse needs to be updated before you can install Designer, a dialog box prompts you to click the URL that takes you to the Designer download site.

2 Select the updates, then click **OK**.

To update from the Welcome page:

1. Click **Help** > **Welcome**.
2. Click the **What's New** icon.



3. Click **New Updates**.
4. Follow the prompts to download and install the latest Designer.

18 Version Control

Designer's version control enables you to do the following:

- Provide simple document management by tracking revisions of your project, along with all the objects and files in that project
- Share those revisions with other members of your team
- Manage the history of your objects
- Make sure that every member of your team is using the same version of your project and Designer

Designer supports the Subversion version control system. Subversion is a stable open source product that is available for no cost and is released under the Apache license. For information about Subversion, see the Apache Subversion Web page (http://subversion.apache.org/). You can also find some pertinent information about using Subversion with Designer in Appendix D, "Version Control with Subversion and Designer," on page 613, as well as "Version Control Best Practices" on page 470.

OpenText Identity Manager introduces package version control using Git in Designer. Git is an open source distributed version control system used to manage, track, and maintain the source code history during software development.For more information, see "Managing Package Versions Using Git" on page 184.

Version control allows teams to work together across continents or just across the hallway, in groups or as a single user. The Version Control view gives you information about changes that your teammates are making in real time. The version control framework allows you to update, merge, and resolve conflicts with your teammates. If you are a single user, version control allows you to make backups, restore older versions, and have the freedom to explore project changes without risking data.

With version control, you can manage the history of your project, and you can go back to a previous revision and create tagged revisions for better release management. Anyone with permission can access these revisions. The Compare Revisions feature allows you to easily scan the history of your project, find relevant changes, and resolve project issues.

Version control functionality is available for all OpenText Identity Manager objects as well as for the contents of the Documents and Toolbox folders, and provisioning objects, but not for Analyzer.

- "Installing a Subversion Server" on page 446
- "Checking In a Project to a Version Control Server" on page 447
- "Importing a Project from a Version Control Server" on page 452
- "Accessing the Version Control View" on page 455
- "Comparing Revisions and Resolving Conflicts" on page 464
- "Version Control Best Practices" on page 470

Installing a Subversion Server

You can either install a Subversion server or use an existing Subversion server. Designer's version control works with all supported Subversion server platforms.

This section provides a quick start for a basic Subversion server on Windows or Linux to use with Designer. For more in-depth information about installing Subversion, see Subversion's installation documentation at Installing Subversion (http://svn.apache.org/repos/asf/subversion/trunk/INSTALL).

- "Downloading and Installing the Server" on page 446
- "Configuring the Server" on page 446

Downloading and Installing the Server

1. Download the most recent version of subversion file:
   - **Linux:** Subversion Packages Web page (http://subversion.apache.org/packages.html)
2. Run the installer and accept the license agreement.
3. Specify the location to install Subversion.
4. For Windows, specify a location in the **Start** menu.
5. Follow the on-screen instructions to complete the installation.

Configuring the Server

1. Create a directory to contain the Subversion server repository.
2. Run the `svnadmin create` command to create the repository at that directory location:

   ```
   svnadmin create [location_of_Subversion_repository]
   ```
3. Go to the [*location_of_Subversion_repository*]\conf directory, which was created when you installed the Subversion server.
4. Edit the `svnserve.conf` file by uncommenting the following lines in the General section (there should be no spaces at the beginning of the lines):

Line to Uncomment	Result
`anon-access = read`	Anonymous users can read your repository.
`auth-access = write`	Authenticated users can edit your repository.
`password-db = passwd`	Usernames and passwords are stored in a file named `passwd` in your `conf` directory.

5. Edit the `passwd` file in the same directory.
6. Remove the sample users from the `Users` section and add your own users.
7. Open a command prompt and start your server by using the following command:

   ```
   svnserve --daemon --root [location_of_Subversion_repository]
   ```

8 Open a second command prompt.

9 Create a `trunk` folder in your repository with the following command:

```
svn mkdir -m "Creating a trunk directory." svn://localhost/trunk
```

10 Authenticate to Subversion.

If your are using Windows, and your username is the same as your Windows username, enter your password. Otherwise, press Enter at the password prompt and enter a username when prompted.

You can also access this server from other computers by substituting `localhost` for the network name of the server machine in the URL.

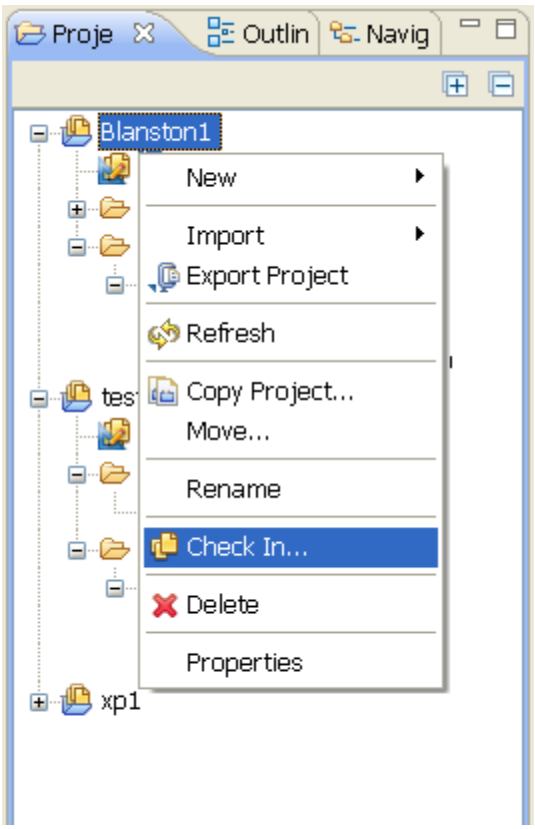
You are now ready to import or add projects to version control by using Designer. You might want to create a more complete directory structure before adding OpenText Identity Manager projects. For more information about how to best use Subversion with Designer's version control, see Appendix D, "Version Control with Subversion and Designer," on page 613.

IMPORTANT: Designer is shipped with the SVN client version 1.5. You can use newer versions of the SVN server, because the SVN servers are backward compatible. However, if you are using the newer version of the SVN server, the client must communicate with the server using the `svn://` or `http://` protocols.

If you create a SVN repository on the local file system using an external client such as Tortoise SVN and then access the SVN repository through Designer using `file:///` protocol, Designer fails to work.

Checking In a Project to a Version Control Server

1 In the Project view, right-click a project name, then click **Check In**.

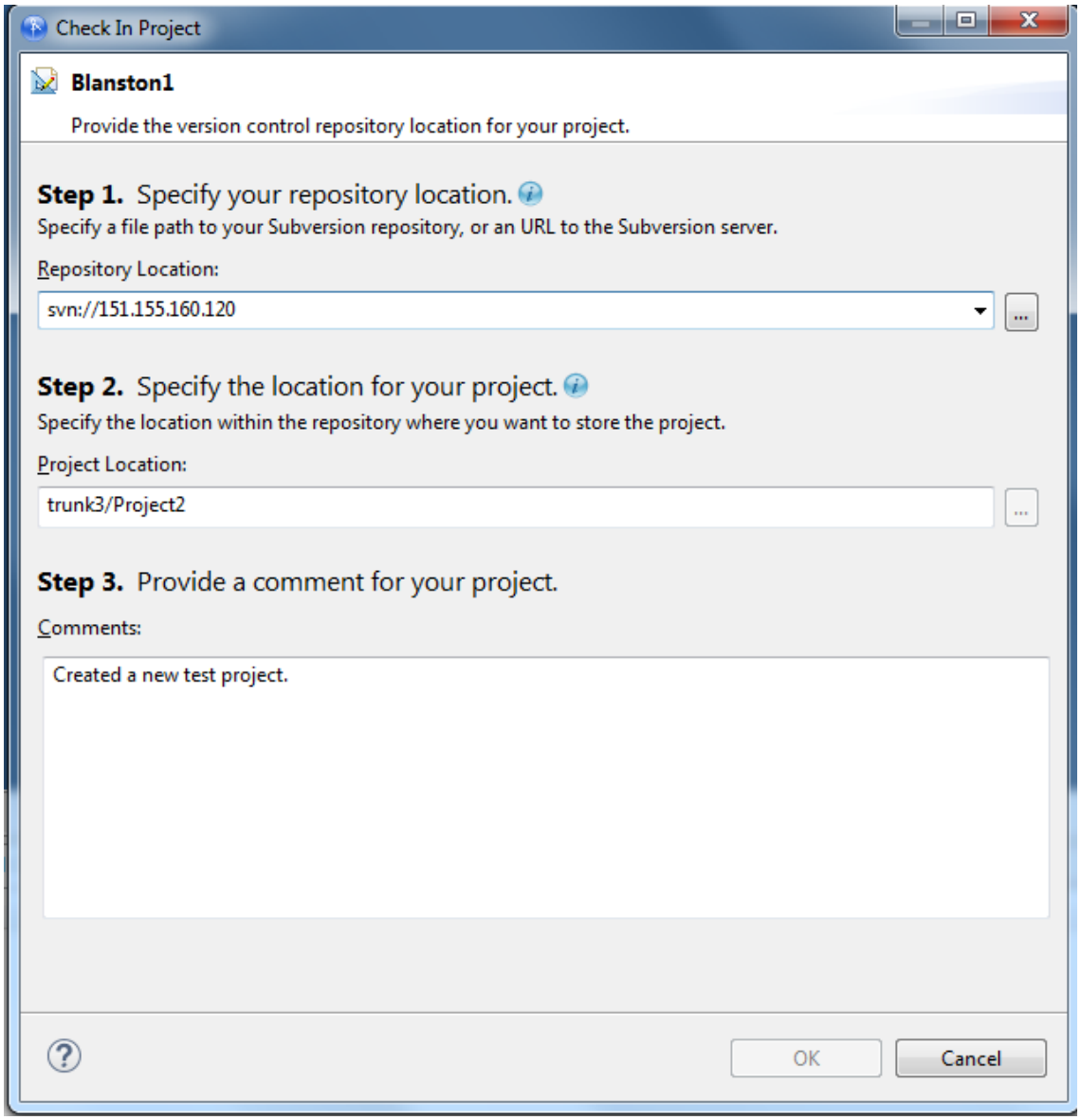


You can also select the **Check Project Into Version Control Server** icon on the main toolbar.

2 If the project you are checking in already exists on the version control server, skip to Step 8.

or

If the project you are checking in does not exist on the version control server, you see the Check In Project page displayed. Continue with Step 3.



3 If you have multiple projects in the Project view and you clicked the **Check Project Into Version Control Server** icon, select the project you want to check into the version control server from the **Select Project** drop-down list. If you select **Check In** from the Project view, you won't see the **Select Project** list.

4 Under **Step 1. Specify your repository location**, provide a URL pointing to where you want the project to reside on the version control server. The Check In Project page gives three examples:

- `c:/subversionrepo`
- `http://subversionserver.mycompany.com`
- `svn://localhost`
- `https://subversionserver.mycompany.com/svn/myrepository`

The list of supported protocols includes:

- svn
- http

- https
- file
- svn+ssh

You can click the **Browse** icon to browse for folders that are saved either locally or on a network drive. You can also create a new folder from the Browse For Folder page.

5 Under **Step 2. Specify the location of your project**, type the folder name that will contain this project on the version control server.

You can also click the **Browse** icon to bring up the Version Control Server Browser page. This browser helps you determine the correct URL where projects are stored and only shows base folders and corresponding projects.

The base folder cannot be a directory of a Designer project. However, the base folder can contain multiple projects as subdirectories. You create base folders through an external SVN client.

6 (Optional) Under **Step 3. Provide a comment for your project**, type a comment concerning the project, then click **OK**.

Whenever you perform an operation that affects the contents of the server, you are prompted for a comment. Comments are useful when keeping track of the changes you make from one session to another.

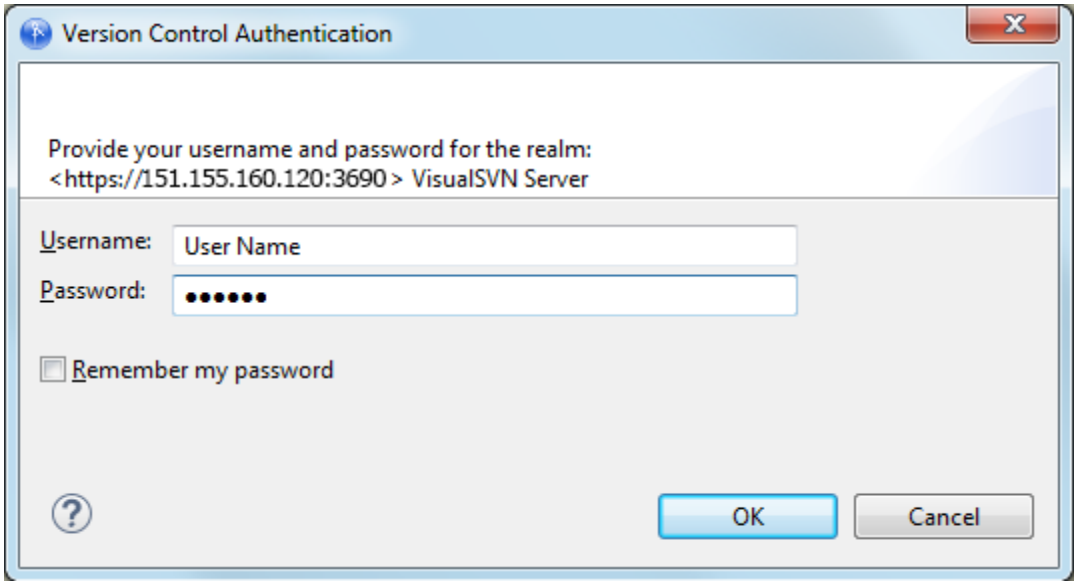
7 (Optional) If you have made changes to more than one project in the Modeler view, you need to save those changes before checking a project into version control.

7a Select **Save All Editors** to bring up the Save Resources page, which allows you to save all open projects.

7b On the Save Resources page, click **OK**. You are returned to the Check In Project page.

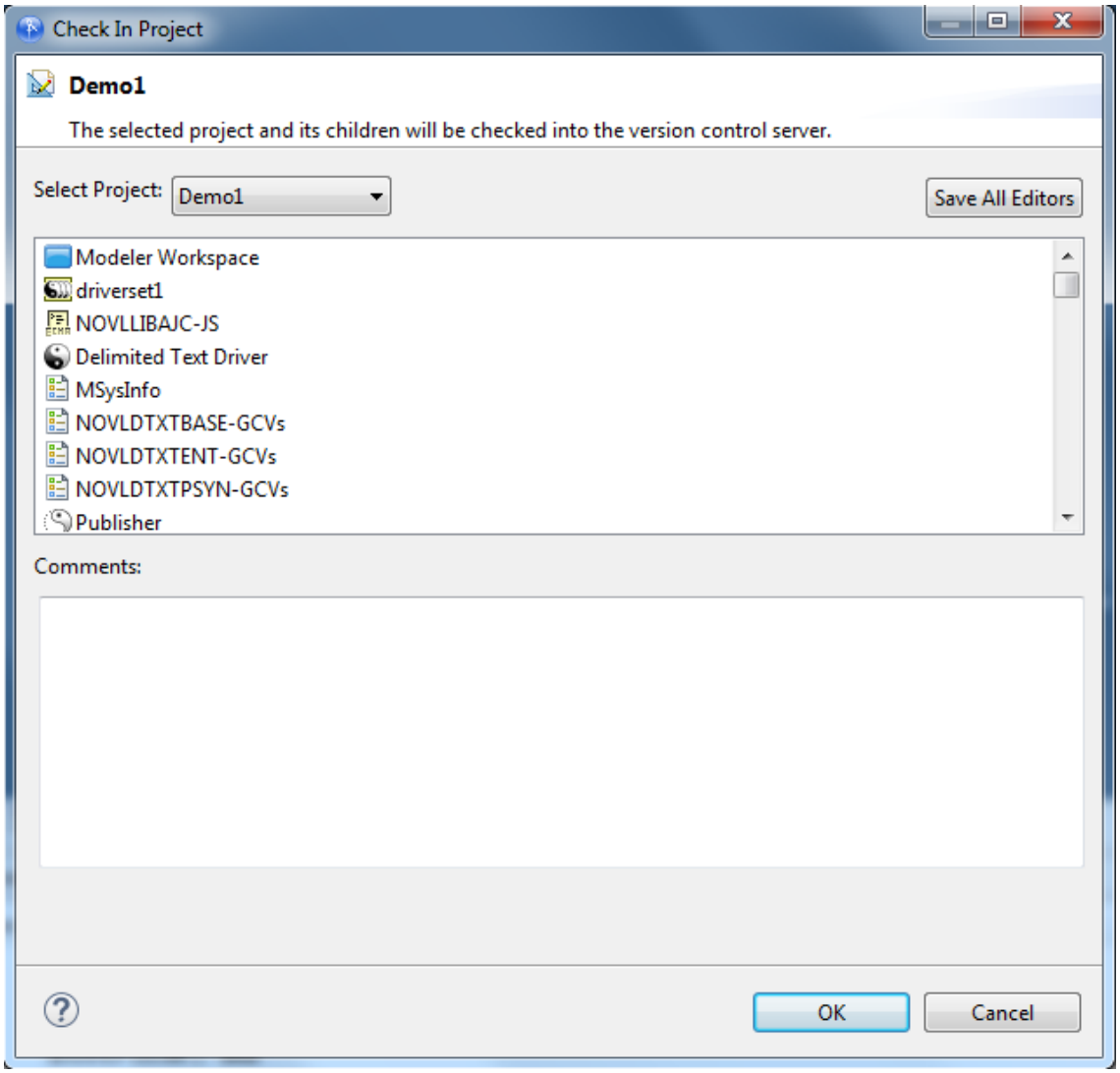
8 Provide authentication to the Subversion server if required.

Depending on the type of security you have set up, you might need to supply SSH authentication, SSL client certificate authentication, or basic HTTP authentication.



9 If you are updating an existing project on the version control server, add new information to the Comment section of the Check In Project page.

If you are updating a project, you see the selected project, the object's children that have changes to be checked in, and objects that depend on the project and need to be checked in. If you choose to check in a single object, you only see that object in the Check-in page.



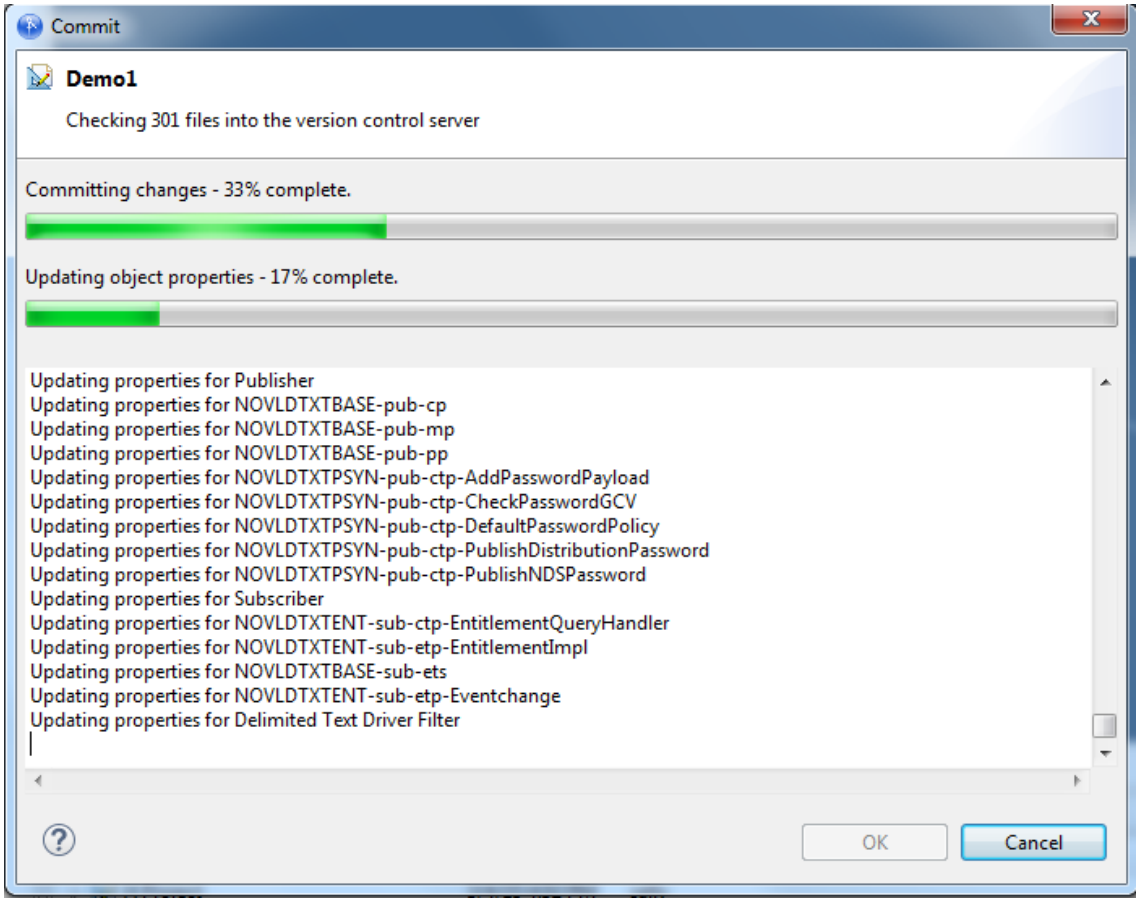
10 If you have more than one project open in the Modeler view, click **Select Project** to choose which project you want to save to version control.

11 (Optional) If you have made changes to more than one project in the Modeler view, you need to save those changes before checking a project into version control.

11a Select **Save All Editors** to bring up the Save Resources page, which allows you to save all open projects.

11b On the Save Resources page, click **OK**. You are returned to the Check In Project page with an updated list of what is being checked in.

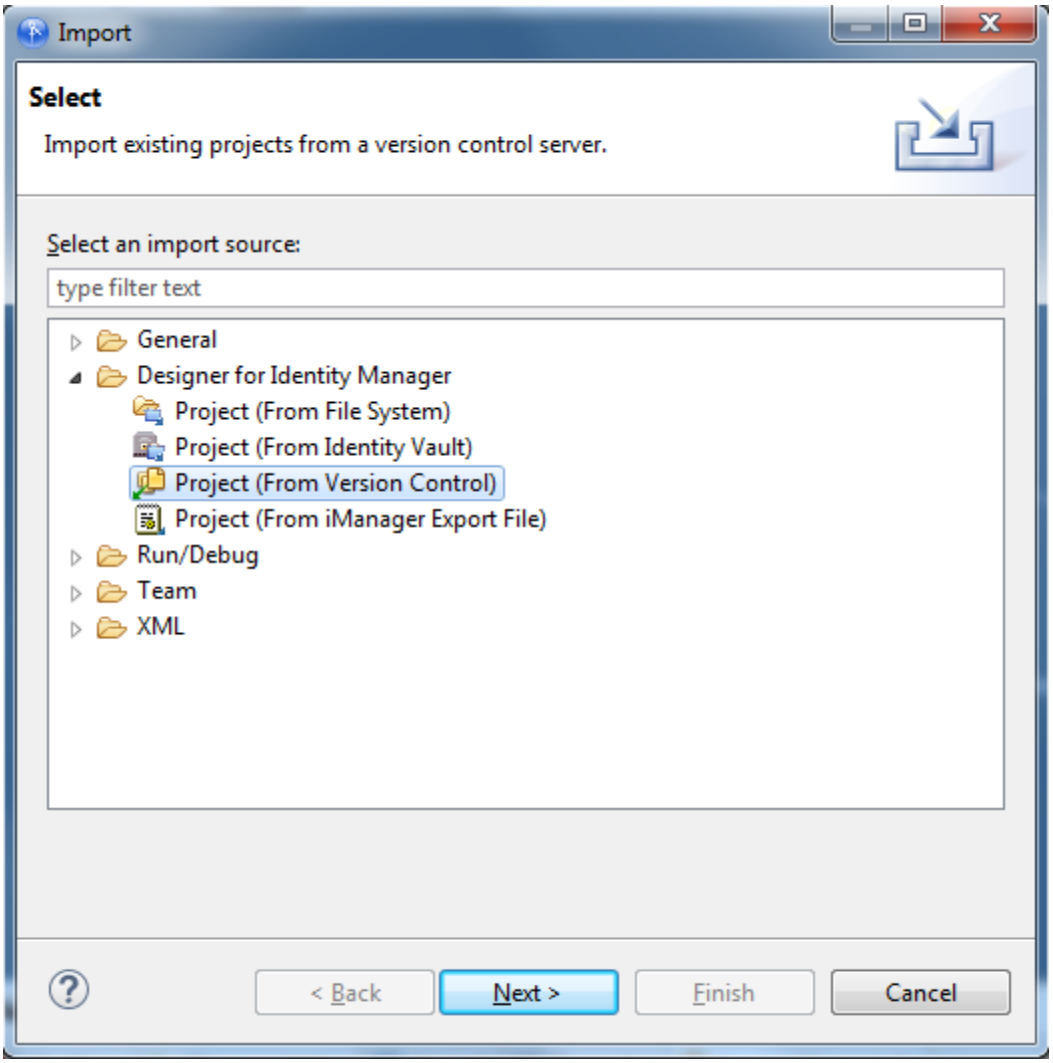
12 Click **OK** to commit the files to the version control server. When the files are committed, click **OK** to close the Commit page.



Importing a Project from a Version Control Server

Designer's Import dialog box lists projects and enables you to select projects that you want to import. There are a number of ways to access the Import dialog box in order to import projects from a version control server, and this example covers one of those methods.

Figure 18-1 *The Import Wizard*



1 In the toolbar, select **File > Import**.

   or

   If no projects are available, select **Import from version control** from the Project view.

2 Click **Project (From Version Control)** > **Next**.

3 Type a URL in the **Version Control Server URL or file path** field, then press Enter. For example:

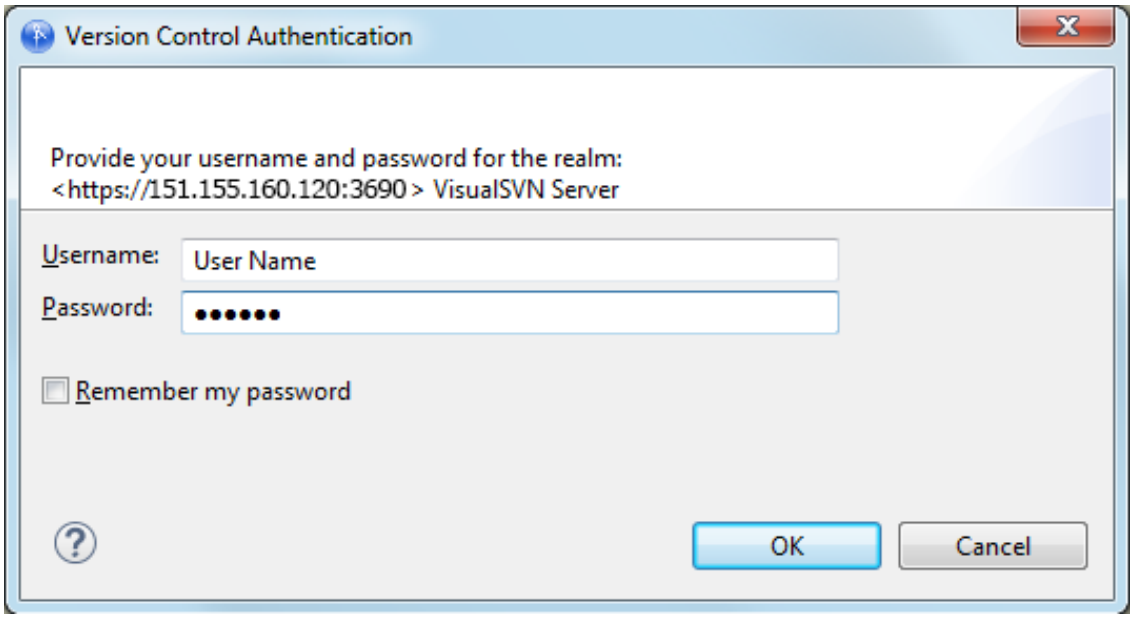
   `https://sun.provo.novell.com/svn`

   `svn://123.123.131.120/trunk`

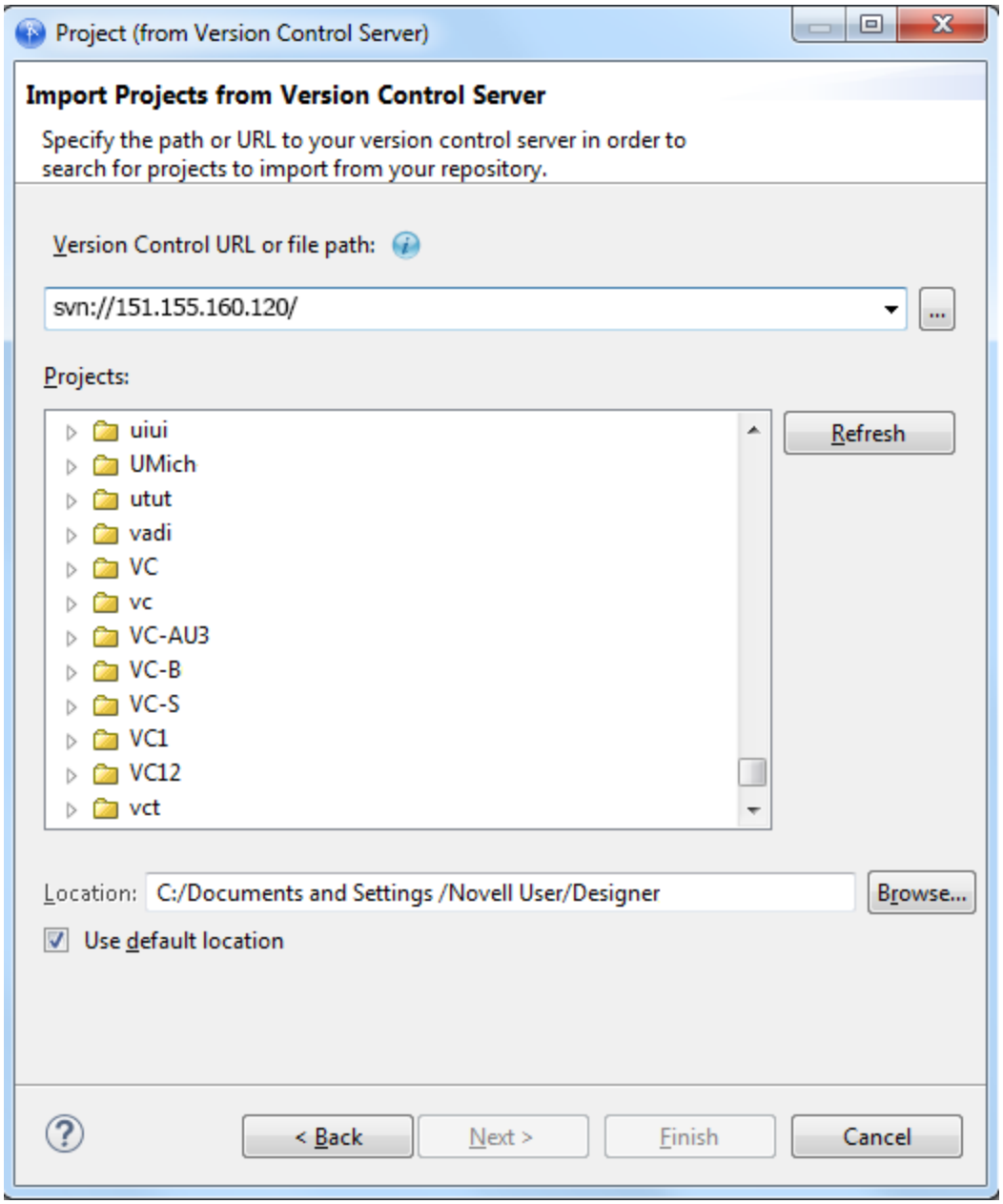
4 (Optional) You can also type a file path to the version control repository, or select the **Browse** icon to browse to the directory where the repository resides.

5 Provide authentication to the Subversion server if required.

   Depending on the type of security you have set up, you might need to supply SSH authentication, SSL client certificate authentication, or basic HTTP authentication.



6 The projects appear under the **Projects**: heading in a tree structure. Select a project file under the directory. Use the **Refresh** icon to see current changes to the repository.



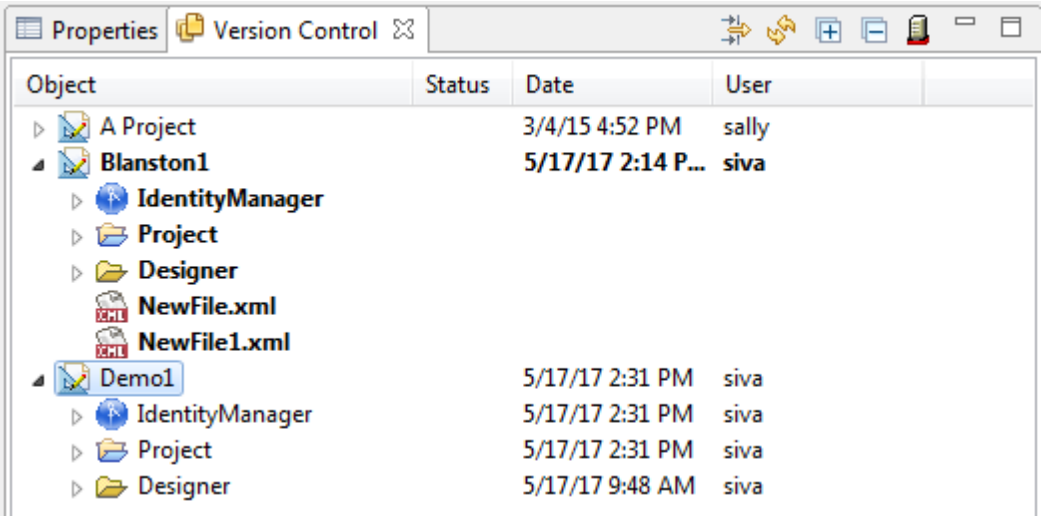
7 Click **Finish**. On the Version Control page that shows you the version control server status, click **OK**.

The projects are imported into Designer and are added to the Project view and the Version Control view.

Accessing the Version Control View

You access version control functionality by using the Version Control view.

Figure 18-2 *The Version Control View*



The Version Control view does the following:

- Gives you a dashboard status of your interaction with version control
- Lists the files that you are working on
- Displays the changes that your teammates have made in real time

The Version Control view is the main interface with version control. You find most of the version control operations and information in this view. This view is empty until you import from or check in a project to the version control server.

The Version Control view automatically displays when you import an existing project from a version control server or check in a project to a version control server. To open the view manually, select **Window > Show Views > Version Control**.

- "Version Control Icons" on page 456
- "Version Control View Headings" on page 457
- "Version Control Options" on page 458

Version Control Icons

The Version Control view contains seven icons that allow you to interact with version control. Six icons are to the right of the Version Control tab. They are the **Filter** icon, the **Refresh** icon, **Expand All** and **Collapse All**, and the **Minimize** and **Maximize** icons. The seventh icon is the **Version Control Project Status** icon, which is located in the bottom right corner of Designer.

Filter Icon: Use the **Filter** icon to limit the number of projects that are displayed in the Version Control view. Click the **Filter** icon, then select the projects you want to filter out of the Version Control view.

Refresh Icon: Click the **Refresh** icon to refresh the Version Control view. Designer communicates with the Subversion server and refreshes the Version Control view with any updates performed by other users who are modifying the same projects.

Expand All/Collapes All Icons: Click the **Expand All** icon to expand all items in the Version Control view. Click the **Collapse All** icon to collapse all items in the Version Control view.

Minimize/Maximize Icons: Click the **Minimize** icon to minimize the Version Control view. Click the **Maximize** icon to maximize the Version Control view.

Version Control Project Status Icon: Mouse over the **Version Control Project Status** icon to see the status of the objects in the Version Control view. The **Version Control Project Status** icon gives you a quick status for version control and works like a traffic light. You can move this icon to a different location in Designer to suit your preferences.

Table 18-1 *Version Control Project Status Icon Colors and Description*

Icon Status	Status Description
	Green. Everything is up-to-date.
	Yellow. Updates are available from the version control server.
	Red. There are conflicts between the local version and the version control server.
	Grey. Designer is unable to contact the version control server.

Version Control View Headings

The Version Control view has four headings: Object, Status, Date, and User.

Object: This column displays the objects that are connected to the project that is stored on the version control server. Right-click an object in the Version Control view to display the available options. These options are covered in "Version Control Options" on page 458.

Status: This column displays the current state for objects in a project, as indicated by the following icons:

Table 18-2 *Status Icons*

Status Icon	Description
(none)	This object is up-to-date, with no new revisions available.
	Unversioned. This object has not been added to the version control server.
	Deleted. This object has been deleted from the version control server.
	Updates with Merge. This object has updates that might conflict with the changes you have made (see "Comparing Revisions and Resolving Conflicts" on page 464).
	The project object has been updated from an older version selected from the Revision History page. The object changes back to normal when you update (see "History" on page 461).
	This object has new child objects available.
	This object has new updates available.
	This object has been modified locally.

Date: This column shows the date when the last changes to the objects in the Version Control view occurred. The date and time change when you modify an object and commit those changes to the version control server.

User: Displays the name of the last person who updated the object.

Version Control Options

Right-click an object in the Version Control view to display the available options.

The options affect the object selected, as well as any child objects that correspond to the selected object. For example, performing a Revert on the project object affects the entire project, but performing a Revert on the Subscriber channel of a Lotus Notes driver only affects the Subscriber channel and any objects (such as policies) that depend on the Subscriber channel.

- "Clean Up" on page 459
- "Commit" on page 459
- "Get Updates" on page 460
- "Revert" on page 461
- "Delete" on page 461
- "History" on page 461
- "Comparing Versions" on page 463
- "Properties" on page 464

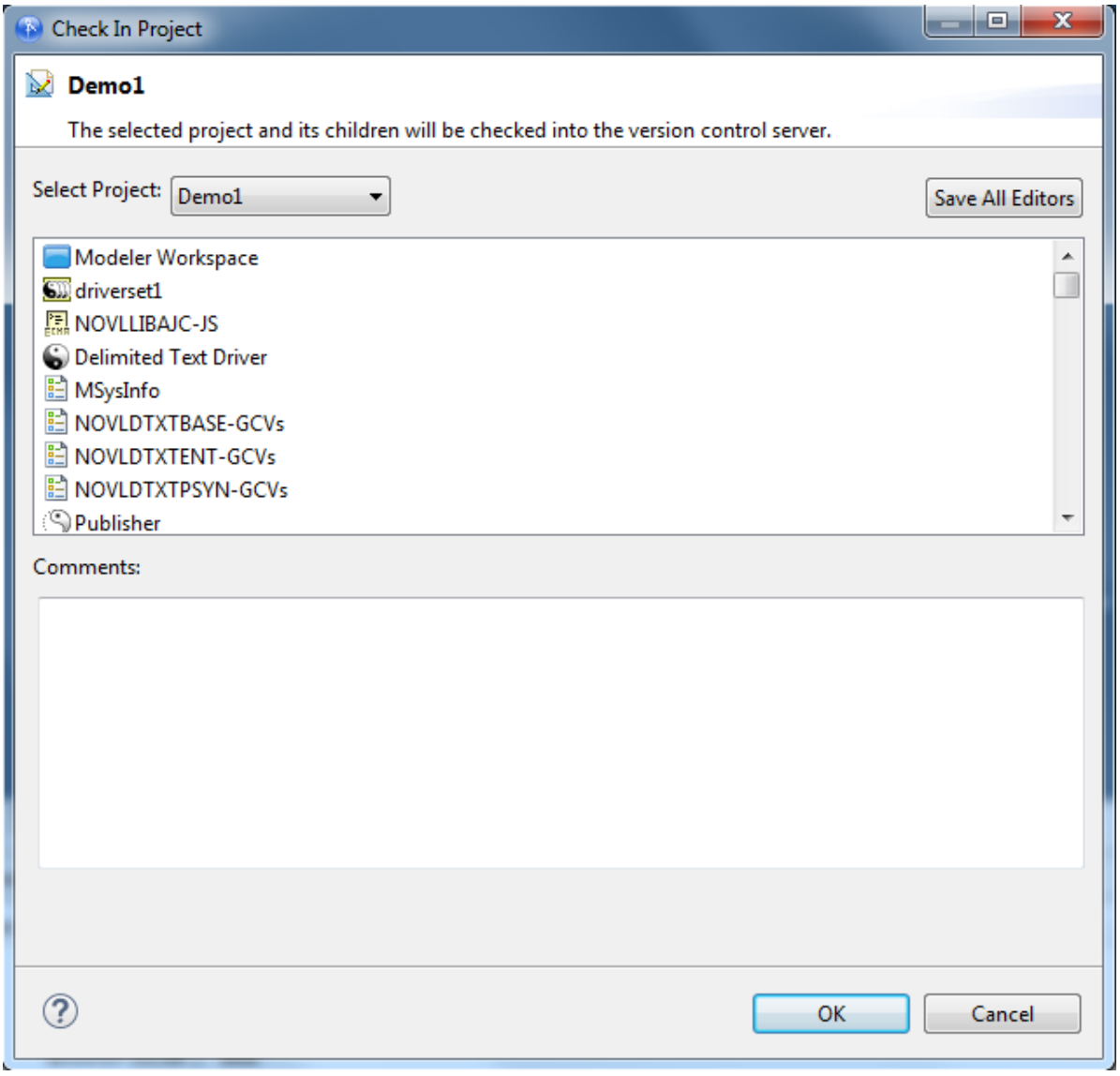
Clean Up

Use the **Clean Up** option only when you are prompted to. Sometimes a project is in a “locked” state. At this point, version control requires you to run Clean Up before it lets you do anything else with the project, and you receive a message telling you to run the **Clean Up** option.

Commit

Use the **Commit** option to have your local changes checked into the version control server for the object you have selected.

Figure 18-3 *Checking In an Object to the Version Control Server*



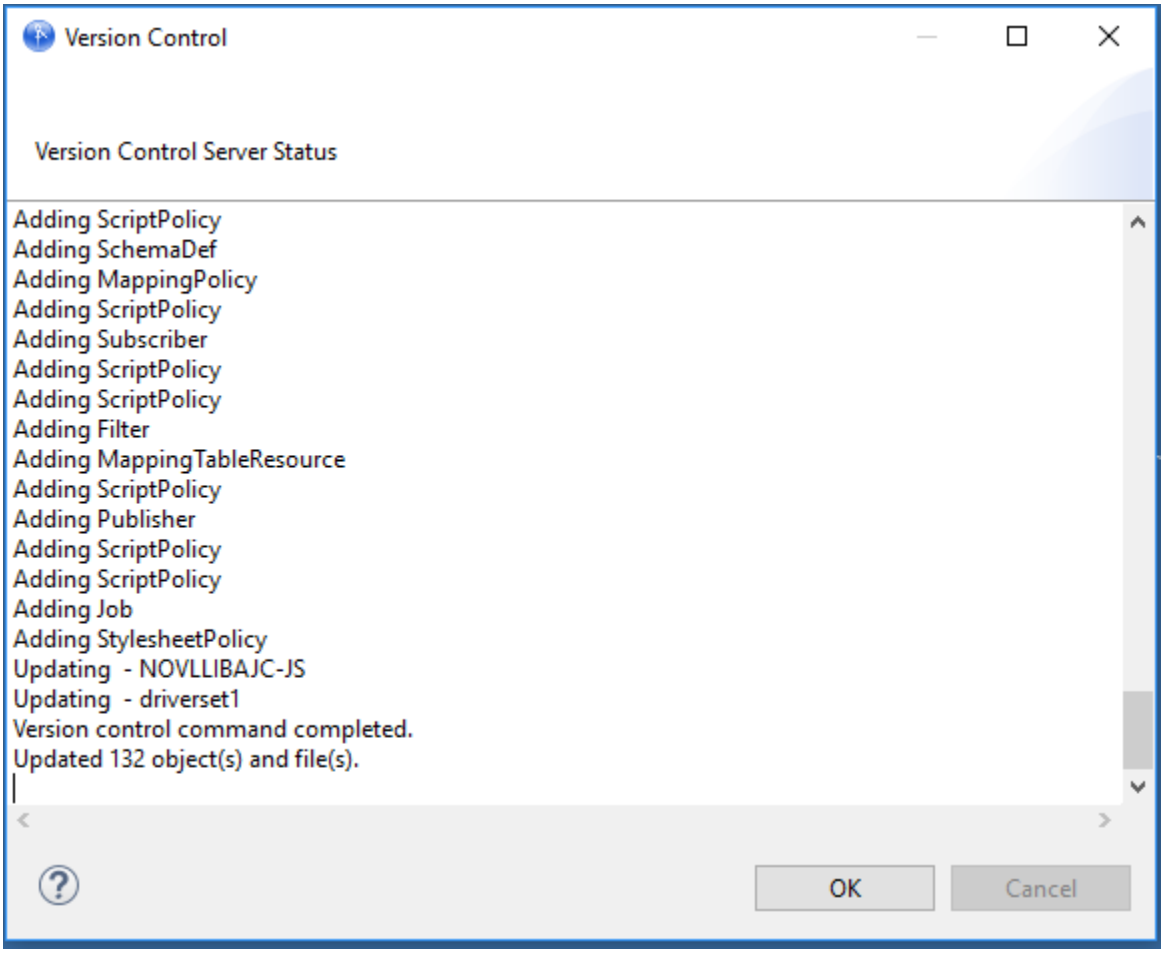
When you click **OK**, the check-in is committed to the version control server. Click **OK** to close the Commit screen.

There are also **Check-in** capabilities in the Project and the Outline view (right-click a project and select **Check In**), and an **Check In** icon in the main toolbar .

Get Updates

Use the **Get Updates** option to get the latest version of the selected object from the version control server.

Figure 18-4 *Receiving Updates from the Version Control Server*



If you have more than one project open that is checked in to the version control server, select which project you want to update from the Update page, then click **OK** to begin the update. If there are conflicts between your local version and the version control server, you see the Conflict Resolution page, which includes a method to resolve those conflicts. For more details, see "Comparing Revisions and Resolving Conflicts" on page 464.

There are also **Update** capabilities in the Project and the Outline view (right-click a project and select **Update**), and an **Update** icon in the main toolbar .

Revert

Use the **Revert** option to return the selected object to the version you last checked out from the version control server. This allows you to cancel your recent changes; you see a message screen displayed, confirming your choice to revert. You can also use this option to restore files that you have deleted since the last time you checked in.

WARNING: By using this option, you lose any changes you have made since the last time you checked it in, including any files in your project that have not been checked into the version control server. Designer deletes all project files that are not in the version control server.

Delete

Use the **Delete** option to delete a project from the version control server. This option is only available for project objects. Although you can delete objects within a project from other views in Designer, you can remove the entire project only through the Version Control view. Selecting the **Delete** option immediately deletes the selected project, and you are prompted for a comment for your actions.

History

Use the **History** option to view the revision history of an object and all the changes that have been made to that object. You can also use this option to select an earlier version of a project.

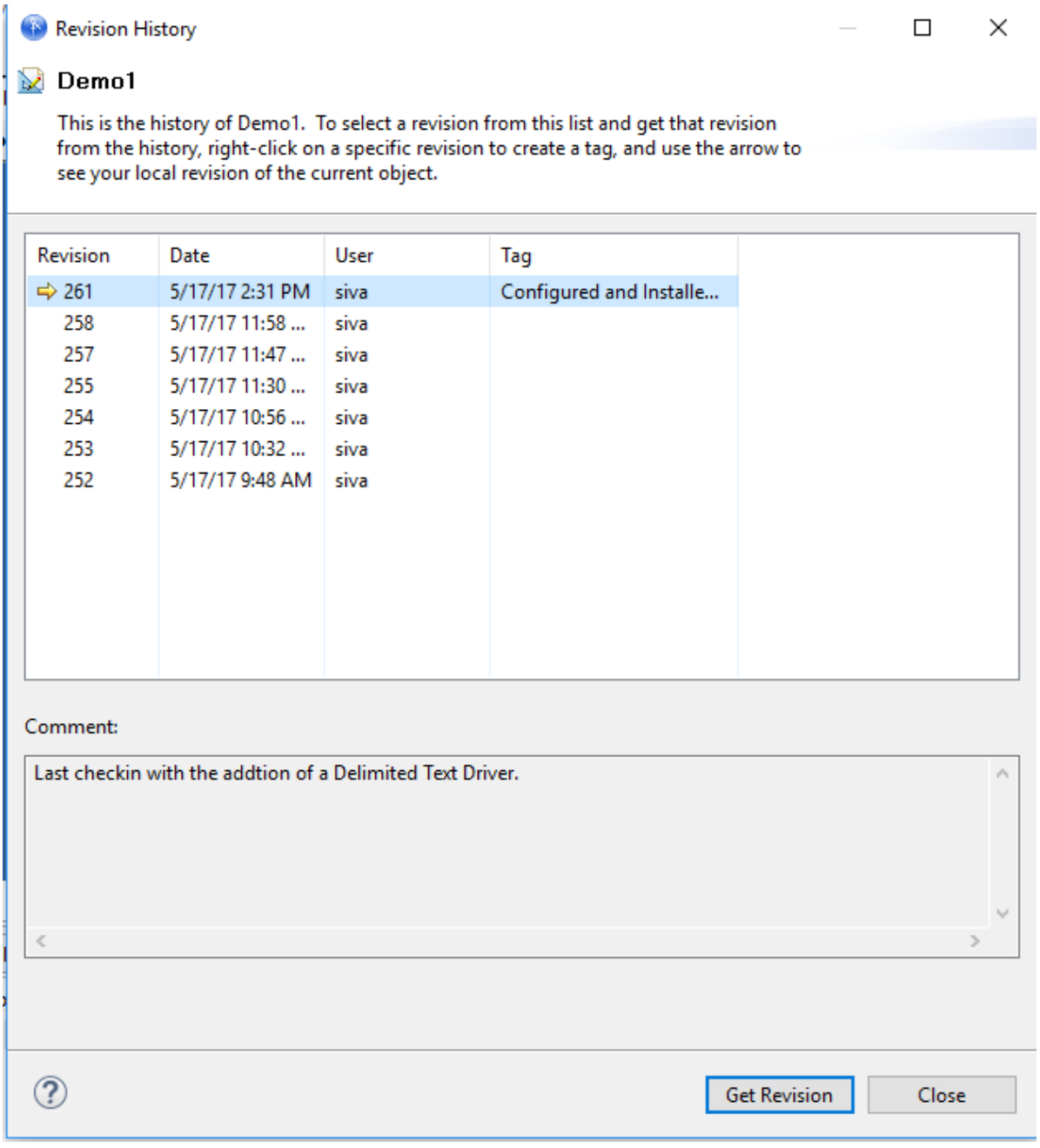
You can use Revision History page to see who made a change, when the change was made, the tag name (if it is filled out), and the comment provided for the change. The yellow arrow indicates your currently loaded version.

Version numbering of projects and how numbering works with the objects in a project is a very complex issue. For more information about how revision numbering works in Subversion, see "How Revisions Work In Subversion" on page 613.

The Revision History Page For a Project

You have more options when you right-click a project object in version control and then select **History**.

Figure 18-5 *Revision History of Projects*



If you select **History** for a project object, the Revision History page allows you to select a version of a project object from the list of revisions. You can then view the contents of earlier versions and bring those versions up-to-date with your latest revision.

Get Revision

Select the revision for the project you want to work with, then click **Get Revision**. Answer **Yes** to save all of the editors in this project. That version of the selected object is downloaded from the version control server and becomes the version of the project you are working on.

If you select an older version of a project, the project has a special status icon in the Version Control view. This icon indicates that your project came from history instead of being out-of-date, but its status returns to normal after you select **Update**.

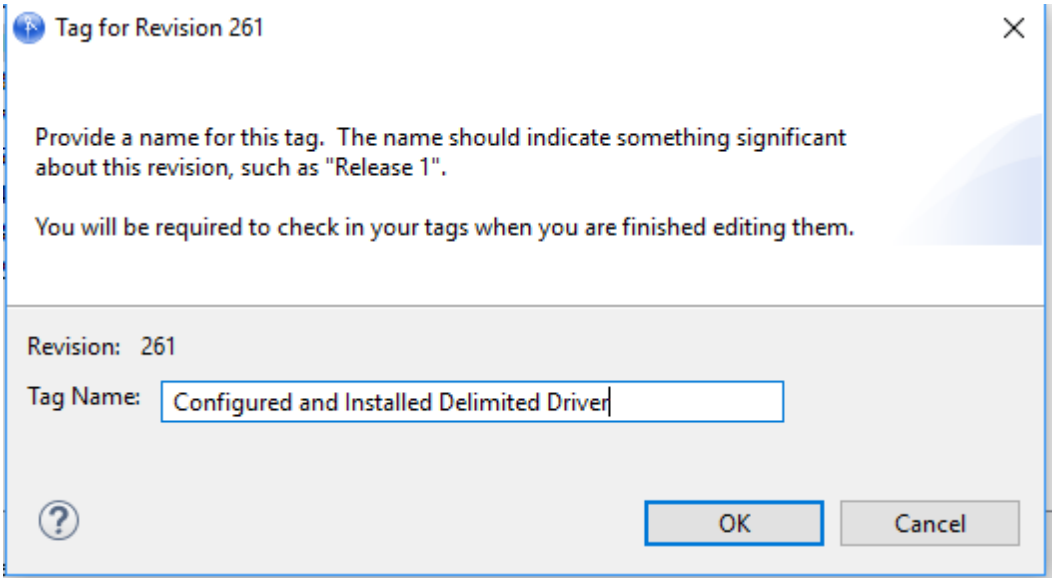
If you make changes to the historical version and select **Update**, you are presented with a Revert Local Changes page, allowing you to keep your local changes or to revert your local changes.

If you have made deliberate changes and want to now save those changes to the version control server, select **Keep my local changes** (default). If you made inadvertent changes to the project, or if you just wanted to see what was in this historical version, select **Revert my local changes before performing the update**.

Creating a Tag for a Project

If you select a project object, you can create a tag for any of the revisions listed in the Revision History page. This allows you to give a revision project a more memorable name instead of a revision number. To create a tag, right-click a revision and select **Create Tag**. This brings up the Tag for Revision page.

Figure 18-6 *Adding a Tag To a Selected Revision*



Provide a tag name that is significant to this version of the object and click **OK**. The tag name is added under the **Tag** heading in the Revision History page. When you close the Revision History page, you are asked to add a comment to all of the tag names that you have added.

Comparing Versions

See “Comparing Revisions” on page 464.

Properties

Use the **Properties** option to view the properties of an object that has been added to version control.

Important information includes the location of the object on the version control server, the loaded revision number, the latest revision number, and any comment concerning the most recent check-in. You cannot make changes to this information.

Comparing Revisions and Resolving Conflicts

- "Comparing Revisions" on page 464
- "Resolving Conflicts" on page 466
- "The Modeler View Layout In a Team-Enabled Environment" on page 467
- "Provisioning Objects" on page 469

Comparing Revisions

Use the **Compare Revisions** option to compare what has changed between your local copy and the latest copy on the version control server. You can compare any object that has been checked in to the version control server. Use this option to compare historical versions to your local copy, or to other historical versions.

NOTE: For the **Compare Revisions** option to work, you must be able to communicate with the version control server. If the version control status icon at the lower right of Designer is grey, Designer is not communicating with the version control server. Mouse over the version control status icon for further connection information.

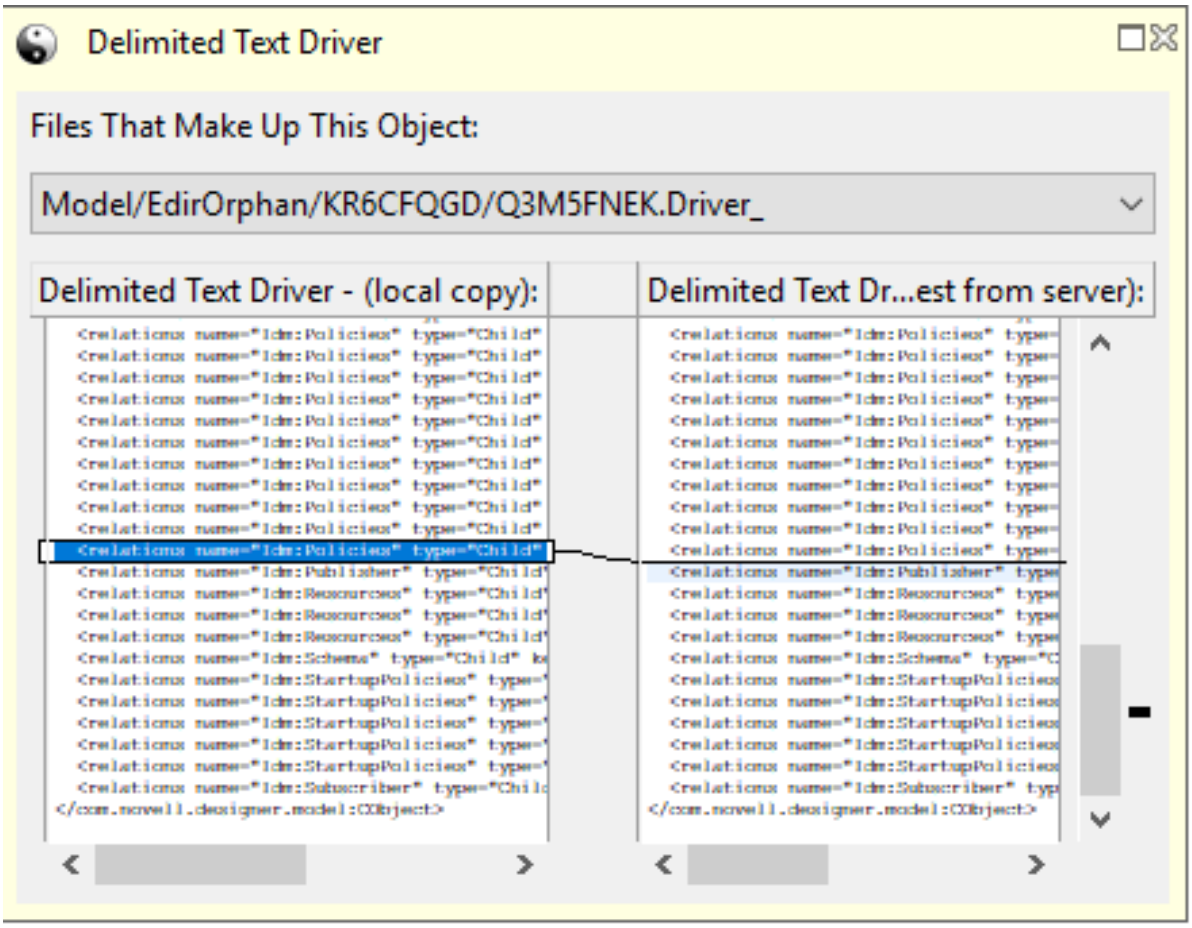
To use the Compare Revisions option, select a project or any other object in the Version Control view and select **Compare Revisions**. The Compare view appears in the main editor section of Designer and is displayed as a tree with the object highlighted. The top bar indicates the object that is selected and which revisions are being compared.

Version control uses a left-to-right display of information. The left side shows local copy information and the right side shows the version from the version control server. Because there is no information in the Outline view, you can double-click the **Compare view** tab to expand the view to fill Designer. Double-click the **Compare view** tab again to have it return to its normal size, or click the **Restore** icon in the lower right corner.

You can select the **Change left-side revision** or **Change right-side revision** icons to view the other versions that you have saved to the version control server. For example, if you want to compare your local copy to a different version on the server, click the right-side icon. If you want to compare the server version to an earlier server version, click the left-side icon. When you select a different version from the History page, the top bar title changes to reflect the different copy comparisons. Click the **Expand All** or the **Collapse All** icon to expand/collapse all items in the Compare view.

To see a snapshot of the changes in an object, click the overview icon to the right of the object to bring up the Overview page for the selected object.

Figure 18-7 *Viewing a Quick Overview of Changes*



If the object you selected is made up of more than one file, you see a drop-down menu listing the files. Select a file from the menu to view the changes to that file.

To view the actual changes in more detail, click the **Expand** icon in the Overview page or double-click the object in the tree view. You can also click the **Compare selected item** icon next to the tree-view icon.

You can use the **Next Difference/Previous Difference** icons or the **Next Change/Previous Change** icons to move between the file's changes. You can also click the blocks on the right side to jump to the file's changes. After you have drilled down and have seen the differences at an object level, click the tree-view icon to return to the tree view.

When to Use Compare Revisions

There are three good reasons to use the **Compare Revisions** option.

- **Finding Problems.** You can use the **Compare Revisions** option to locate when a specific problem was introduced to a project. You can determine when a change was made, who made that change, and why the change was made. If someone on your team broke a policy, you can see when it was broken, who broke it, and what their comment was when they checked it in.

- **Change Overview.** You can also use the **Compare Revisions** option to get an overview of the changes that have been made to a project. By choosing different revisions, it is easy to see all of the changes that were made to a project in a given period of time.
- **Conflict Resolution.** The **Compare Revisions** option can help you resolve conflicts. When you compare your local version and the latest from the server, the conflicts are highlighted in red and you can see the specific conflicts. See "Comparing Revisions and Resolving Conflicts" on page 464.

Resolving Conflicts

- "Example 1: Checking In Changes to the Same Object" on page 466
- "Example 2: Core Model Object Conflicts" on page 466
- "Example 3: Deleted Projects" on page 466

Example 1: Checking In Changes to the Same Object

If Bob and Terri are working on a project and they both try to edit the object in the version control server at the same time, they have a conflict.

Suppose Bob checks in first. Designer is communicating with the version control server in the background and collects status information about all of the objects that are checked out. If there is a conflict, the **Version Control Project Status** icon changes to red and Terri sees a warning message when she mouses over the icon.

When Terri attempts to check in, she receives an error message telling her to update before she checks in.

If she clicks **OK** and performs the update, version control tries to automatically merge the differences between Bob's and Terri's changes. However, if their changes cannot be automatically merged and Terri tries to update, she sees the Resolve Conflict page, allowing her to see the differences between her local version and the version on the version control server.

The red markers on the right side of the Resolve Conflict page show the data that is in conflict, and the blue markers show the modified local data. Terri can then choose to either keep her local version or to overwrite her local version with the one on the version control server. The Resolve Conflict page also shows the path of the file with the conflict.

Example 2: Core Model Object Conflicts

In some conflicts, the core model objects can merge manually at an attribute level, allowing you to change the attributes so that they are no longer in conflict. If the conflict is of this nature, you see the Conflict Resolution page, allowing you to manually resolve the conflicts.

When you have made the necessary attribute changes, select **Resolve Conflict**.

Example 3: Deleted Projects

If the project has been deleted from the version control server, you are given three choices: delete the local project, keep the local project as an unversioned project, or restore the project on the version control server.

The Modeler View Layout In a Team-Enabled Environment

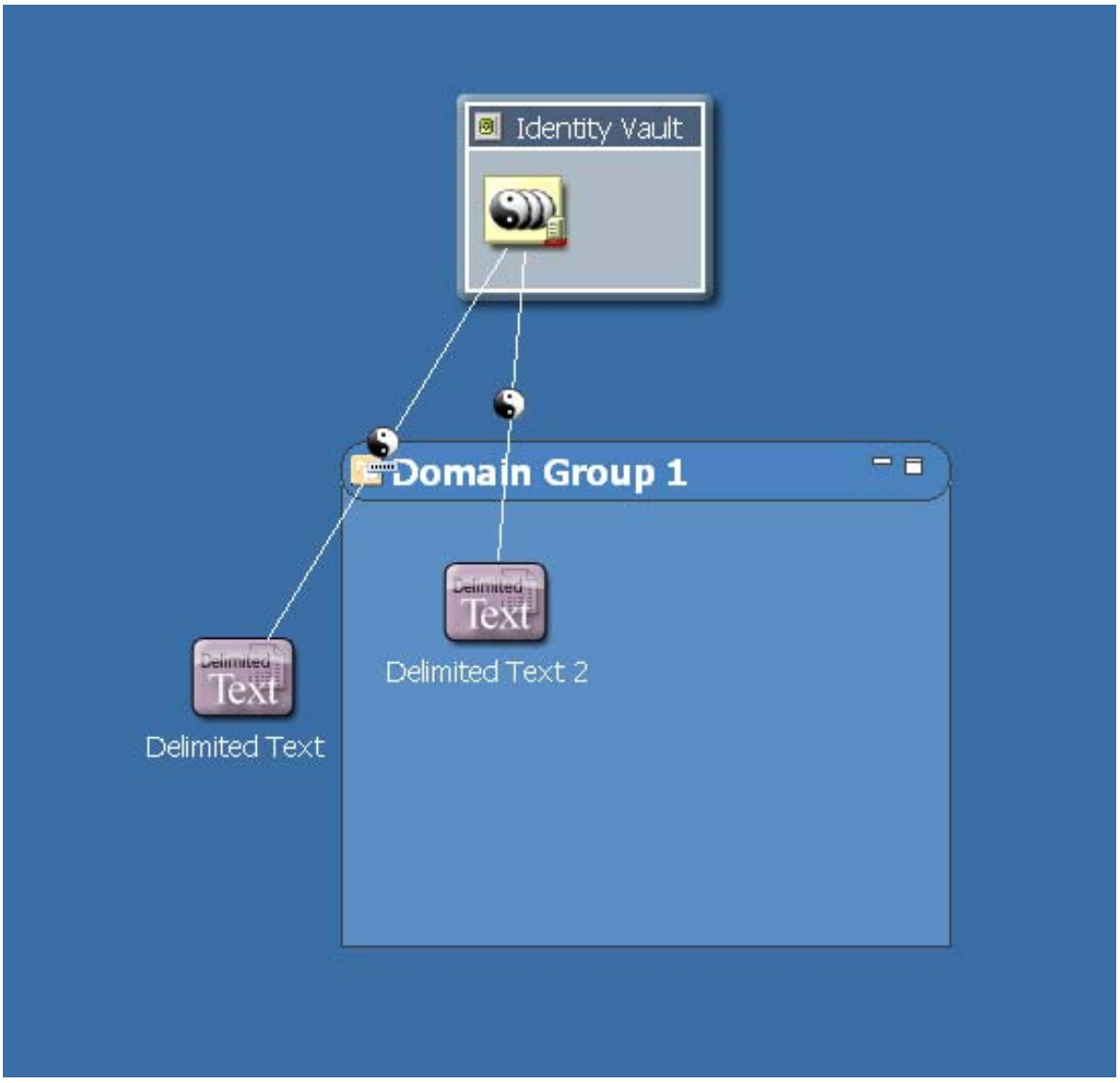
Designer handles saves by multiple users in a complex manner. Your personal Modeler view layout in a team environment changes as others change their Modeler view layout and check in their changes to the version control server. When you perform an Update from the version control server, you get the last Modeler view layout that was checked into version control. Remember that it's just the layout that is changing and not the data.

For example, suppose Bob and Terri are working on a new project. Terri creates the project and checks the project into version control.

Terri tells Bob about the new project and Bob imports the project from the version control server. Bob then adds a domain group and another driver, and checks those changes into version control.

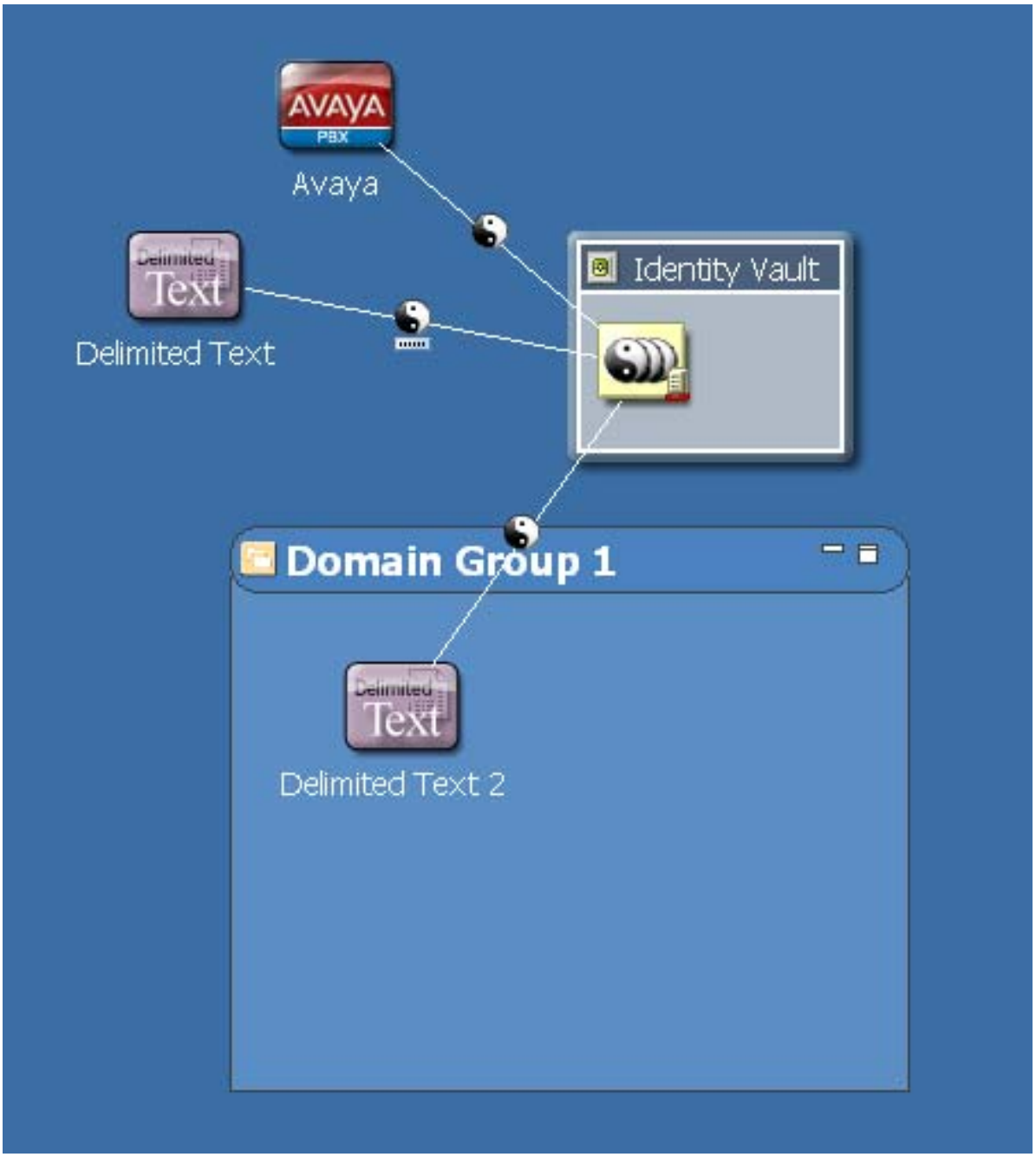
During this time, Terri was working on the first driver and made only minor changes to the Modeler view, but they were enough to create local differences. When Terri saves her changes locally, then updates the project from the version control server, she sees that her Modeler view changes are merged with Bob's Modeler view changes.

Figure 18-8 *Terri's Modeler View Changes are Merged with Bob's*



However, if Bob changes the Modeler layout again (and checks in) and Terri does not (no conflict), Terri gets Bob's Modeler layout the next time that Terri updates from the version control server.

Figure 18-9 *Bob's Last Check-in*

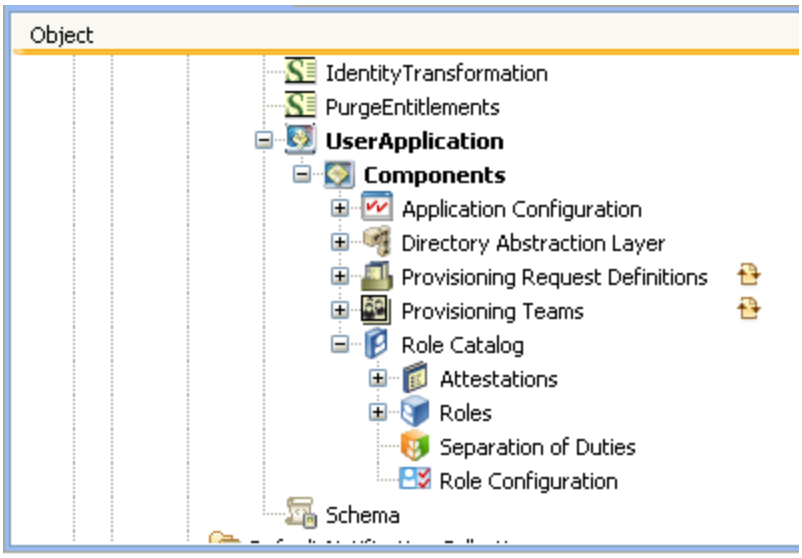


As a best practice, define a Modeler layout that the team can live with and leave it alone.

Provisioning Objects

In Designer 3.0 and above, provisioning objects such as the directory abstraction layer, Provisioning request definitions, teams, and roles, can all participate in version control. The Version Control view below illustrates how provisioning objects appear.

Figure 18-10 *Provisioning Objects in the Version Control View*



The Version Control view reflects provisioning objects in a slightly different hierarchy than the Outline view. Under the **User Application** entry, you see a node called **Components**. This is the main node under which all provisioning objects are located. **Application Configuration** and **Locale Configuration** are also new nodes in the tree. System objects and unsupported objects are also visible in the Version Control view.

Version Control Best Practices

Managing a team environment with version control can be a challenging task. Combining version control with Designer has its own set of issues. This section includes some tips and best practices for using version control with Designer.

- "Best Practices" on page 470
- "Managing Packages Best Practices" on page 471
- "Best Practice Scenarios" on page 472
- "Subversion and Version Control Interaction Rules" on page 478

Best Practices

- **Coordinate all Designer upgrades with your entire team.** When you upgrade to a new version of Designer, many of the files in your project are changed by the project converter, so you need to coordinate with the rest of your team. In the ideal upgrade process, everyone checks in all of their changes, one team member runs the project converter and checks in the converted project, then everyone installs the new version of Designer and re-imports the project.
- **Coordinate deployment.** When you are using version control and the same OpenText eDirectory server with multiple people, it is possible to overwrite changes. You should coordinate deployment with your team members to make sure that you do not overwrite other team members' changes. Best practice is to assign one person to deploy a project to a production environment.

- **Assign policies.** Assign one team member to a policy rather than having multiple team members work on one policy. Multiple team members writing and modifying shared policies in a driver is a recipe for disaster.
- **Define an acceptable Modeler layout for the team.** Personal Modeler layouts in a team environment are only maintained if there is a version control conflict on the Modeler layout between your Modeler view layout and another's Modeler view layout. If there is no conflict and you perform an update from the version control server, you get the last Modeler layout that was checked into version control.
- **Compare, Check in, and Check out the objects at the root level** . This helps to ensure that all objects are stored in the version control repository.
- **Check in the project from the version control view for existing projects** . You can check in from the outline view or project view as well, but it may cause performance issues.
- **Use the same version of Designer within the team when working with version control** . This is because the newer version of Designer may create objects that the older version of Designer may not be able to process.
- **Update your Identity Vault before migrating from a test environment to a production environment** . Change the IP address and the credentials of the Identity Vault to point to the production OpenText eDirectory server before you migrate the test OpenText eDirectory shared servers to the production environment.
- **Use a production environment administrator account that is located in the production server network** . It is recommended to have the production environment administrator on the same network as the production server to avoid network or VPN issues. This is because, importing or deploying of designer projects to OpenText Identity Manager can be slow over VPN.

Managing Packages Best Practices

This section includes some best practices for managing packages in version control.

- "Creating Packages" on page 471
- "Checking In and Updating Packages" on page 472
- "Upgrading and Downgrading Packages" on page 472

Creating Packages

- A single user should be assigned to create a package and its newer versions, and then check in the packages to enable the other team members to add or modify the content of the packages.
- A single user should be assigned to create a driver and check in the corresponding packages of the driver.
- A Designer project cannot contain multiple instances of the same package. When you import or create packages in a version control environment, ensure that you do not import and then check in the same package and version already checked in by another user. Multiple instances of the same package, especially a common package used by more than one parent package or driver, can cause conflicts in Subversion.

Checking In and Updating Packages

1. Check in the entire catalog, and check in the driver and the parent objects of the driver (if available).
2. Update the entire catalog.

   This ensures that all the objects are imported into the Designer workspace.

Upgrading and Downgrading Packages

A single user must be responsible to upgrade and downgrade the packages and check in.

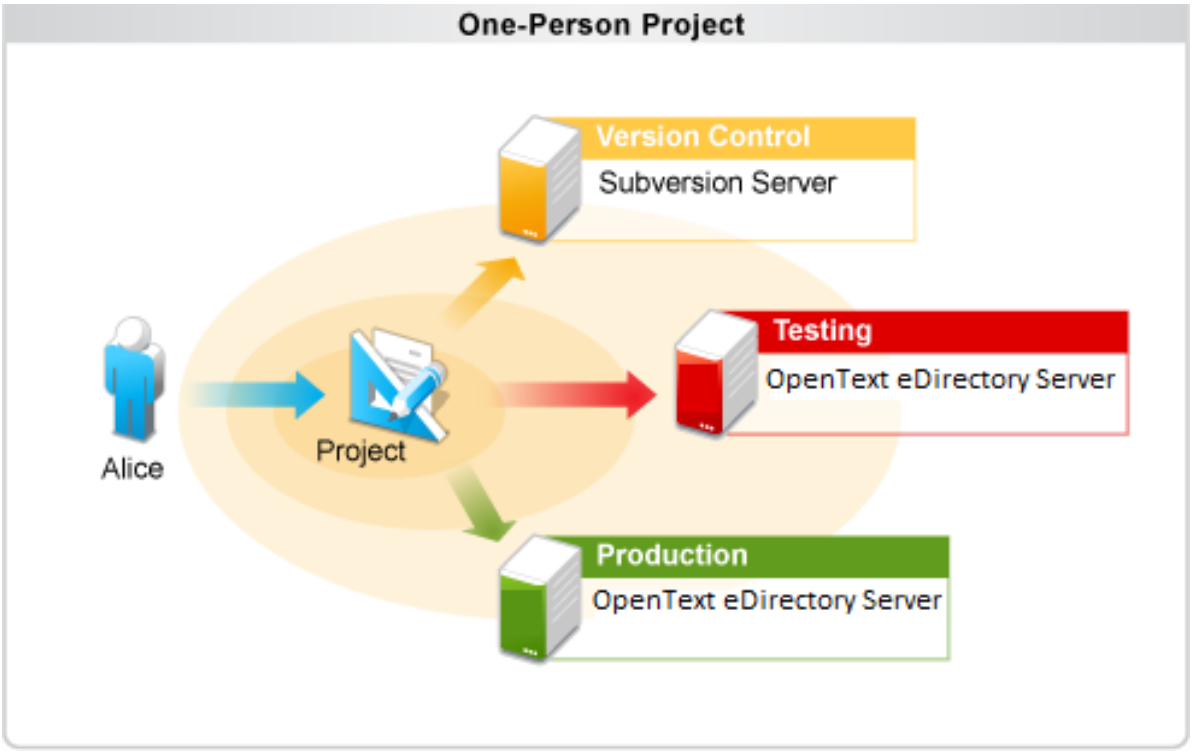
Best Practice Scenarios

There is no one-size-fits-all scenarios for using version control with Designer. This section identifies some user situations that we used for best practice scenarios. These scenarios are specific step-by-step guides to be used in addition to those outlined in the Best Practices section.

- "One-Person Project" on page 473
- "Small Team with One Shared OpenText eDirectory Server" on page 474
- "Small Team with Individual OpenText eDirectory Servers" on page 475
- "Medium-Sized Team with a Shared Test and Production Environment" on page 476
- "Single Consultant Working for Multiple Companies" on page 477

One-Person Project

Figure 18-11 *One-Person Project*



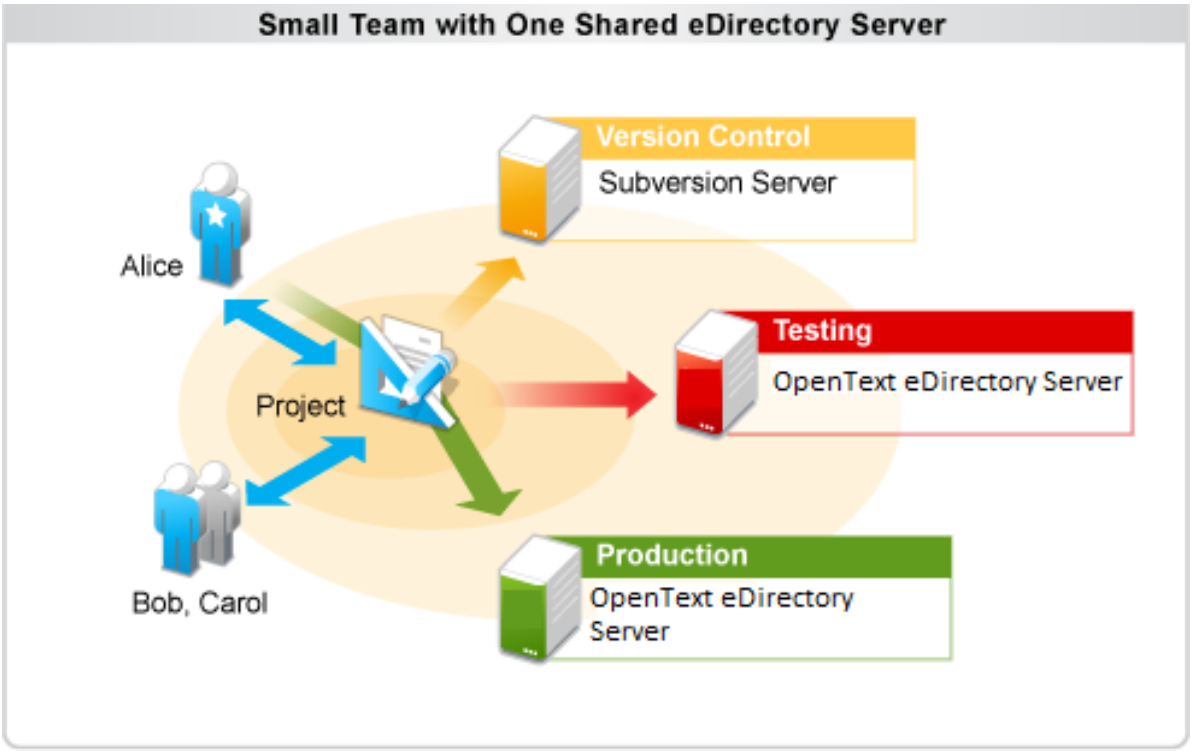
Version control is very useful in a team environment, but it is also very useful in an individual environment. Version control allows a single developer to make backups, restore older versions, and have the freedom to explore project changes without risking data.

Alice decides to work on a project alone. She creates a new project and checks that project in to the version control server. She makes changes to the project and deploys them to a development server for testing. She frequently checks her changes into the version control server so she can easily explore the history of her project later.

Alice can optionally use tagging to specify which project revisions are stable revisions. If she is unsatisfied with any project changes, she can revert those changes or get an older copy of her project from history. When she is happy with her changes, she deploys the project to an OpenText eDirectory server in the production environment.

Small Team with One Shared OpenText eDirectory Server

Figure 18-12 *Small Team Scenario #1*



Alice, Bob, and Carol are working together on a project. They are assigned the following roles:

- Alice - Administrator
- Bob and Carol - Engineers

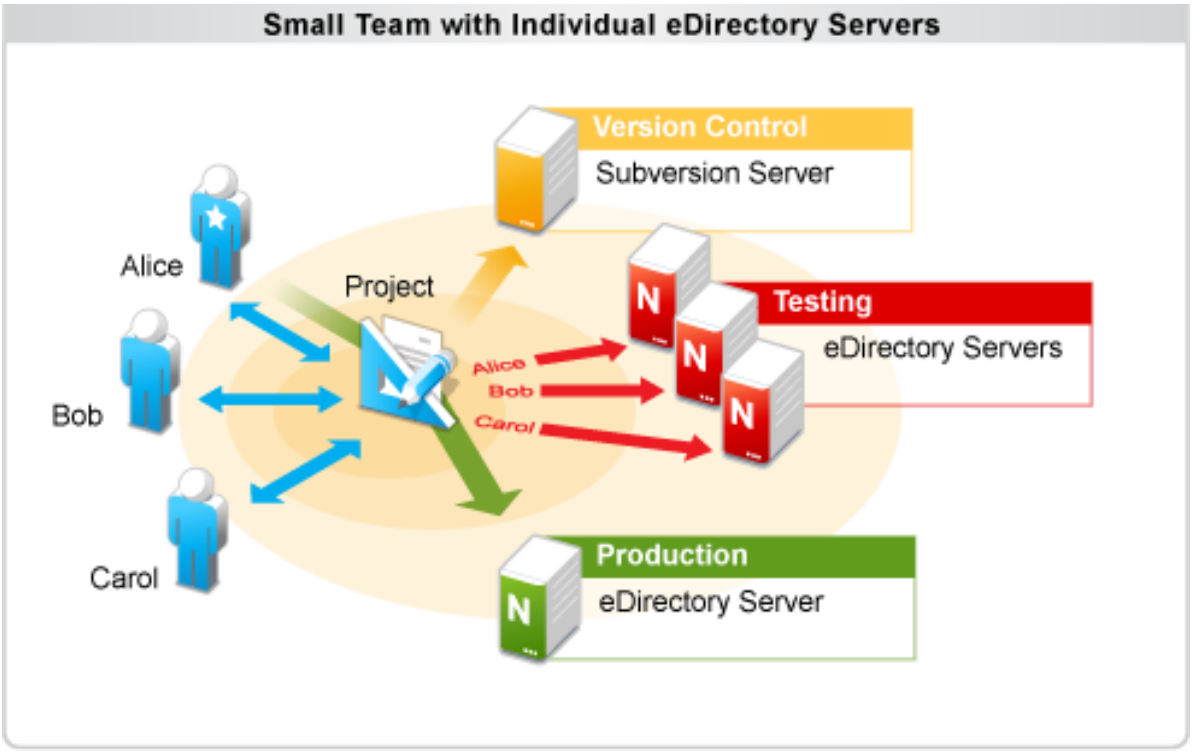
Alice creates the new project and checks it into the version control server. Bob and Carol import that project and they all work on the project together. Alice, Bob, and Carol agree on ownership of OpenText Identity Manager objects and do not often edit each other's objects. When Alice, Bob, or Carol want to deploy their changes to the shared development environment, they are careful to deploy just their own changes and not corrupt or overwrite the common objects that can overlapped during development. Everyone is diligent about updating frequently in order to avoid conflicts.

They all deploy to the same shared development server so they can test their changes in the same environment. When each team member is happy with the results, they check in their changes to the version control server.

When they are ready to deploy their project to an OpenText eDirectory server in the production environment, Alice performs an update to get the latest changes from the version control server and then deploys the project to the production server. Alice manages all deployment to the production server so the team maintains control over the changes in the production environment.

Small Team with Individual OpenText eDirectory Servers

Figure 18-13 *Small Team Scenario #2*



Alice, Bob, and Carol work together on a project. They are assigned the following roles:

- Alice - Administrator
- Bob and Carol - Engineers

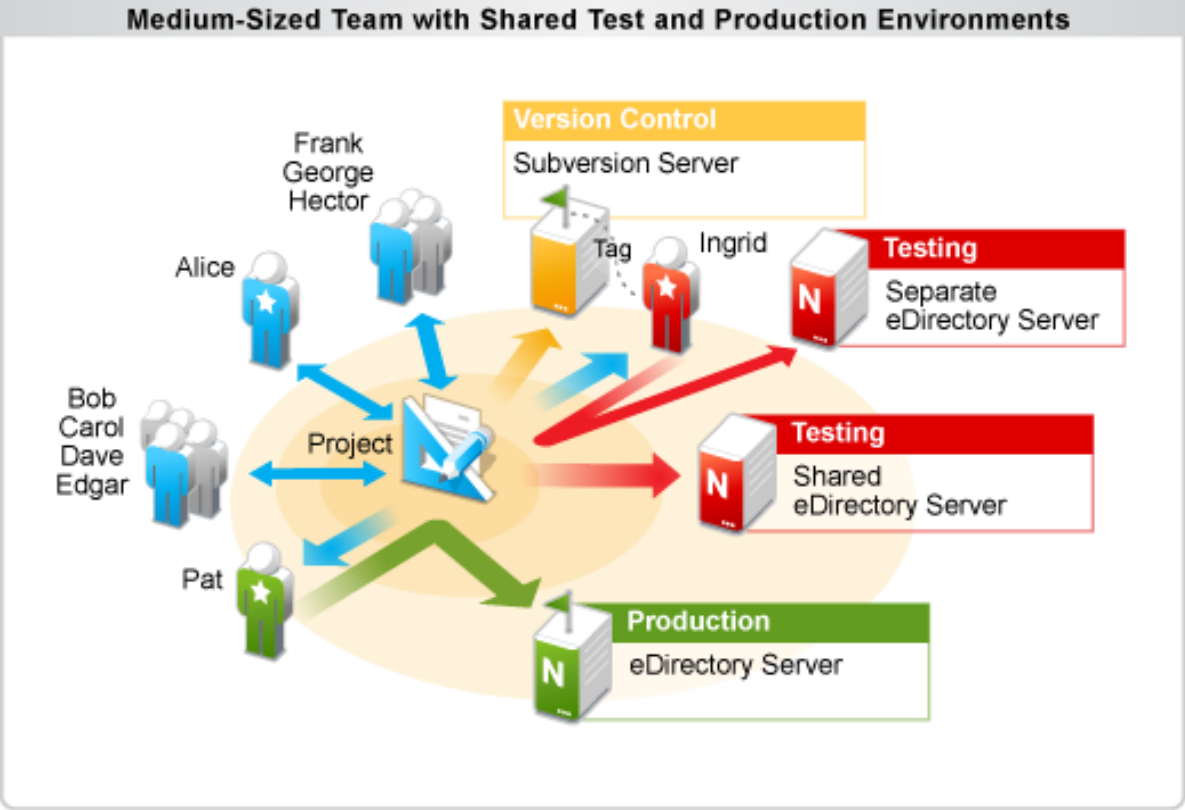
Alice (the administrator) creates a new project and checks it into the version control server. Bob and Carol then import that project and they all work on the project together. Alice, Bob, and Carol don't need any boundaries for object editing and they are all welcome to edit every object in the project. They update frequently and resolve conflicts when they occur.

Alice, Bob, and Carol each have their own OpenText eDirectory development server to deploy to and can deploy changes without the need to consult each other. They change, deploy, and test their changes and then check them into the version control server.

When they are ready to deploy to the production server, Alice updates her project to get the latest changes from version control and then deploys them to her development server. After she has verified that everything works as expected, she deploys the changes to the OpenText eDirectory server in the production environment. Alice manages all of the deployment to the production server to make sure it is a controlled environment.

Medium-Sized Team with a Shared Test and Production Environment

Figure 18-14 *Medium Team Scenario*



Alice, Bob, Carol, Dave, and Edgar all work together on a project. The following roles are assigned to all team members working on this project:

- Alice - Administrator
- Frank, George, and Hector - Part time consultants
- Bob, Carol, Dave, and Edgar - Engineers
- Ingrid - Integration Test Engineer
- Pat - Production Environment Administrator

Frank, George, and Hector work part-time on this project and consult for other projects. Alice (the administrator) creates the project and checks it into the version control server. Bob, Carol, Dave, and Edgar import the project from the version control server and they all begin working on the project and deploying to the same OpenText eDirectory development server.

Frank, George, and Hector work mostly in an advisory capacity and do not own any objects in the project. They consult with Alice before making changes. Frank, George, and Hector are careful when they deploy changes so that they don’t overwrite the changes of the object owners.

Alice, Bob, Carol, Dave, and Edgar mostly focus on changing their own objects, but Ingrid (the integration test engineer) focuses on testing the entire project on a separate development server. She imports the project from version control and updates frequently to get changes from the rest of

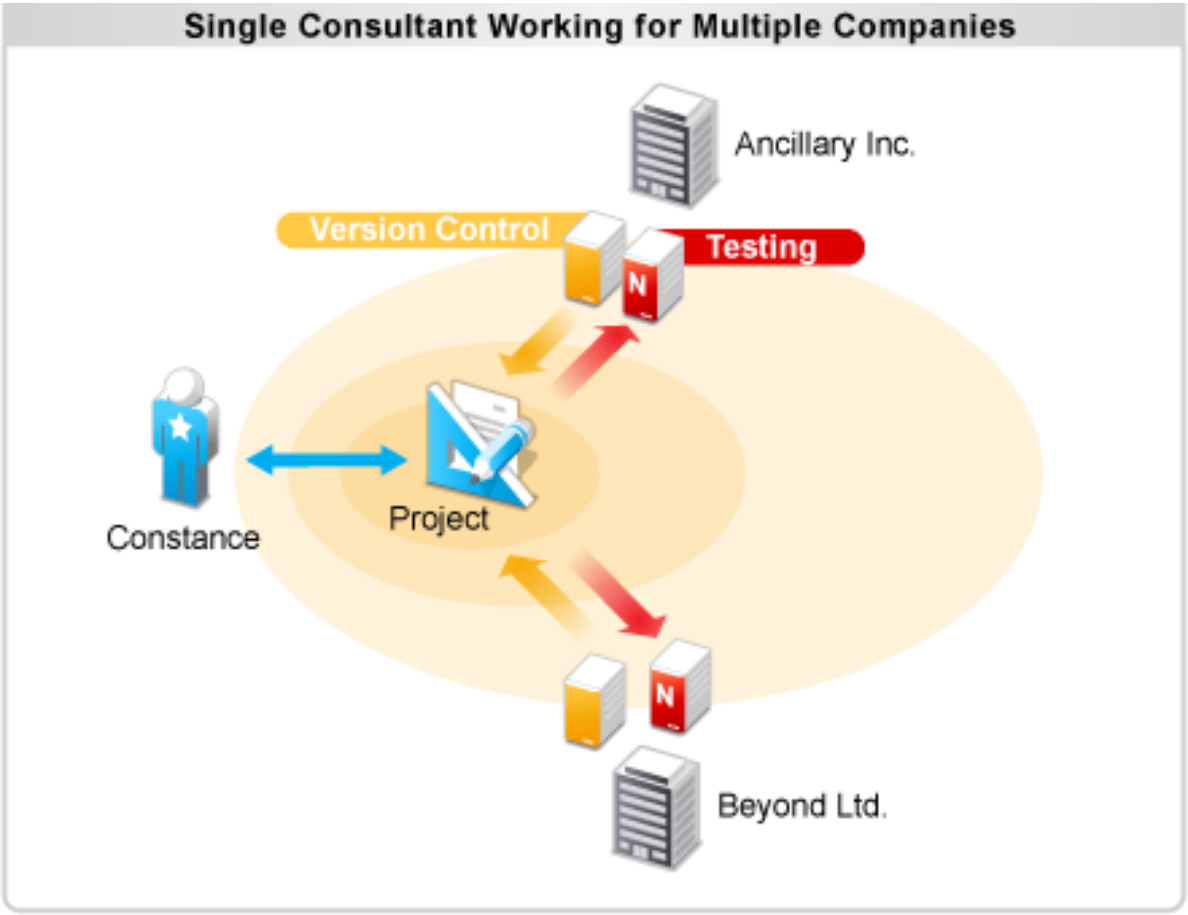
the team. She deploys those changes in the controlled development environment and tests them there. Ingrid makes only the changes necessary to deploy to the test server and does not check any changes into the version control server.

When Ingrid is satisfied with a version of the project, she creates a project tag in version control and certifies that revision of the project as deployable to the production environment. She then asks Pat (the production environment administrator) to deploy the project to the production server and tells him which tag should be deployed.

Pat imports the project from the version control server. He then uses the Get from History function to get the specific revision that Ingrid has tagged. After he has that version, he makes only the changes necessary to deploy the project to the production server and deploys the project. The rest of the team can continue to work on the project during this time because Pat has locked his version of the project to the revision that Ingrid has certified as deployable to the production environment.

Single Consultant Working for Multiple Companies

Figure 18-15 *Working for Multiple Companies*



Constance (the consultant) works for multiple companies, helping them with their OpenText Identity Manager projects. On Monday, she works for Ancillary Incorporated. She imports the project from the version control server at Ancillary Inc. and deploys the project to the Ancillary development

server. Constance communicates frequently with the Ancillary Inc. team members and makes sure to never overwrite the objects from the Ancillary Inc. team on the OpenText eDirectory production server.

On Tuesday, Constance works for Beyond Limited. She closes the Ancillary project and imports the project from the Beyond Limited version control server. She follows established procedures when working with the Beyond Limited team and carefully separates the changes for each company.

Subversion and Version Control Interaction Rules

- **Do not use the Subversion command line.** People familiar with the Subversion command line might be tempted to use it with Designer to perform simple commits or updates. Designer has many tools to manage the merging and object dependencies within an OpenText Identity Manager project. Using the Subversion command line bypasses these tools and can easily lead to a corrupted project and data loss.
- **Do not use other Subversion clients.** Tortoise, Subclipse, or any other Subversion client can cause the same problems as the Subversion command line. Do not use them on the same working copy you are using for Designer.

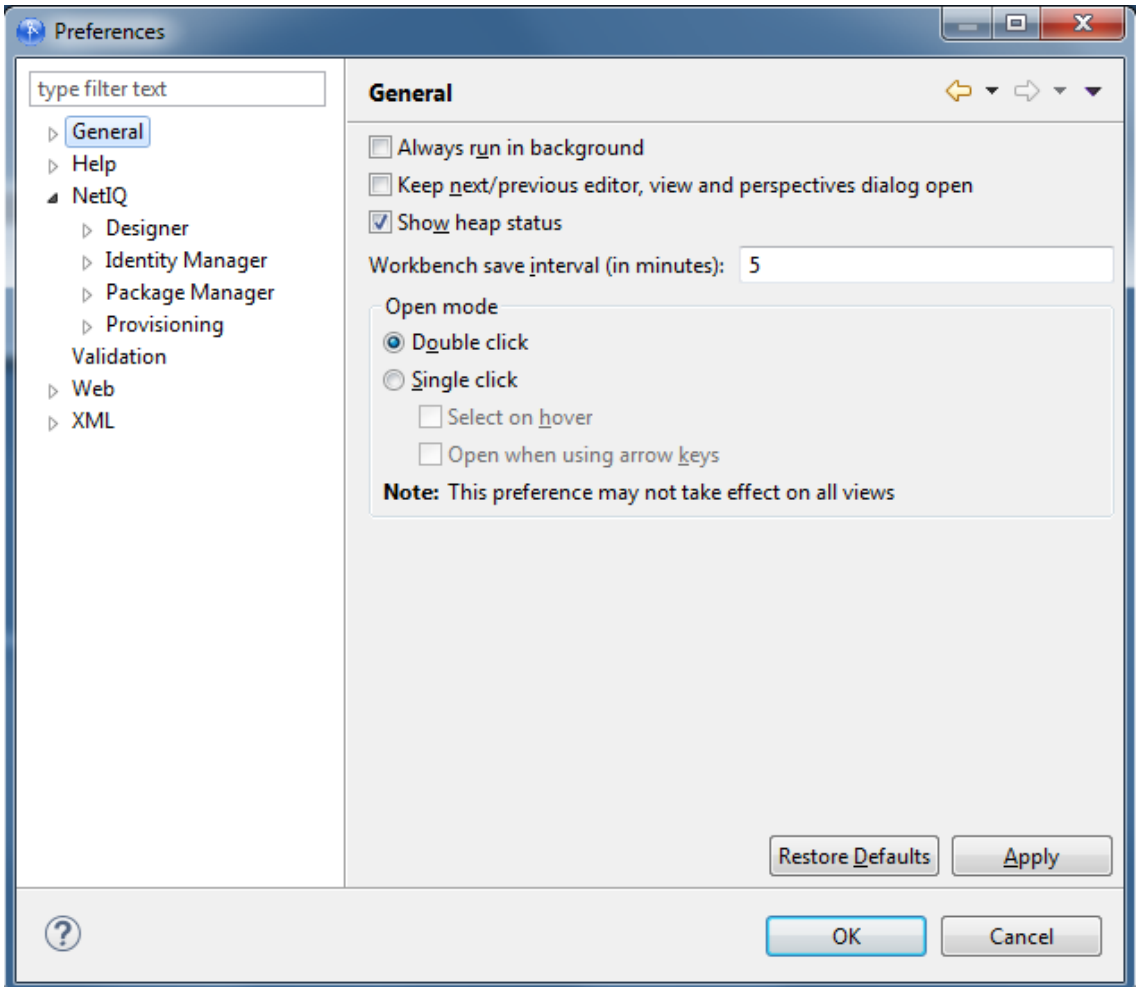
19 Setting Preferences

- “Finding Preference Pages” on page 479
- “General” on page 480
- “Help” on page 491
- “NetIQ” on page 491
- “Validation” on page 518
- “Web” on page 518
- “XML” on page 522

Finding Preference Pages

You customize Designer by setting options in Preferences.

1 From the main menu, select **Window** > **Preferences**.



2 Select a heading or navigate to a subheading.

3 Make changes, then click **Apply** or **OK**.

IMPORTANT: If there are no projects in Designer, the Preference page does not appear when Designer is loaded or started. The page is displayed when a project is created in Designer.

General

The General preferences page includes the following settings:

Table 19-1 *Preferences: General*

Setting	Description
type filter text	Searches all the preferences and shortens the tree view, depending upon what you type in the edit box.

Setting	Description
Always run in background	Enables operations to run in the background without disturbing you.
Keep next/previous part dialog open	Keeps the editor and view dialog boxes open when an activation key is released. Normally, the dialog box closes as soon as the key combination is released.
Show heap status	Places a field in Designer's bottom right corner and displays the amount of memory being used of total memory available.
Open mode: Double click	Opens a project when you double-click it.
Single click: Select on hover	Selects the setting when the cursor hovers there.
Single click: Open when using arrow keys	Opens the setting when you select it.

Additionally, the following preferences categories appear as General sub-pages:

- "Appearance" on page 482
- "Compare/Patch" on page 484
- "Content Types" on page 485
- "Editors" on page 486
- "Keys" on page 488
- "Network Connections" on page 488
- "Perspectives" on page 489
- "Startup and Shutdown" on page 489
- "Web Browser" on page 490
- "Welcome" on page 490

Appearance

Figure 19-1 *Preferences: General > Appearance*

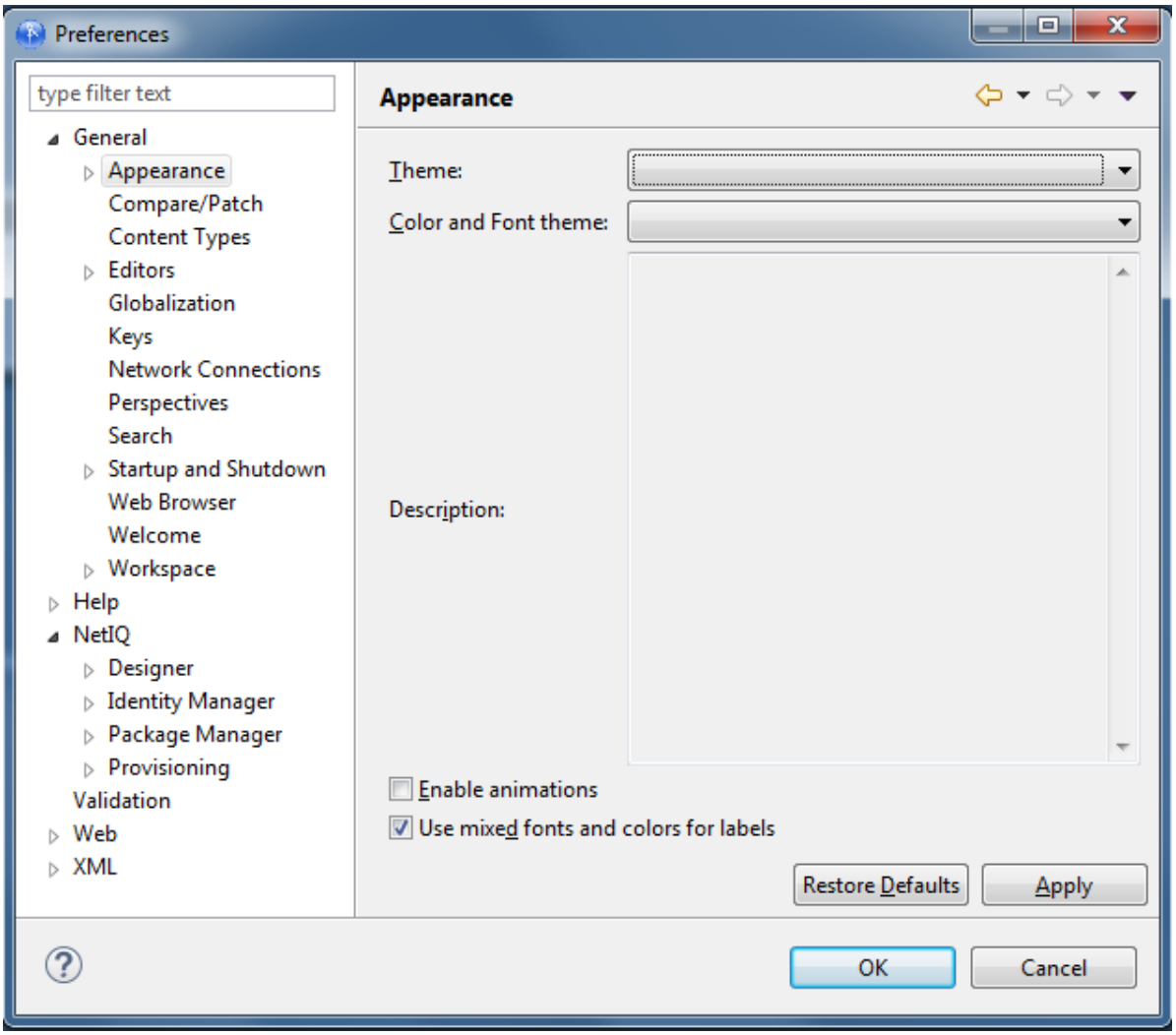


Table 19-2 *Preferences: General > Appearance*

Setting	Description
Current Presentation	Allows you to choose between Designer's presentation, the current presentation, or the Eclipse 2.1 style presentation.
Override presentation settings	Alters how the tabs and views appear in the workbench
Editor tab positions	Positions tabs on the Modeler, NetIQ XML editor, or Text editor at the top or bottom.
View tab positions	Positions view tabs (for example, the Project view tab) at the top or bottom of views.

Setting	Description
Perspective switcher positions	Positions the Perspective Switcher at the left, top left, or top right of the workbench.
Show text on the perspective bar	Determines whether text (for example, Designer) displays next to the icons in the Perspective Switcher.
Current theme	The general theme (colors and fonts) that Designer uses. Choices are Default (current), reduced palette, and R 3.0 theme.
Show traditional style tabs	Displays square Windows-style tabs. The alternative is rounded tabs.
Enable animations	Animates views (for example, Fast Views) and editors that you minimize, maximize, or restore. Reinforces tasks in Designer.
Enable colored labels	Displays colors on labels, if the labels have colors defined.

Colors and Fonts

To change a color:

1. Under **General,** expand **Appearances**.
2. Select **Colors and Fonts**.
3. Expand an option (for example, **Basic**).
4. Select an item (for example, **Active hyperlink text color**).
5. Click the color button.



6. Select a color from the Color palette, then click **OK**.

To change a font:

1. Under **General**, expand **Appearances**.
2. Select **Colors and Fonts**.
3. Expand an option (for example, **Basic**).
4. Select an item (for example, **Banner Font**).
5. Click **Change**.
6. Select a font, style, and size, then click **OK**.

Label Decorations

Label decorations display additional information about an item on its label or icon. Select the desired label decorations:

- Binary Plug-in Projects
- File Icons Based on Content Analysis
- Java Method Override Indicator
- Java Type Indicator
- Linked Resources
- Provisioning Element Decorator

Compare/Patch

This Eclipse functionality customizes the behavior of the comparison editor. When you select to compare or synchronize two or more resources in the Workbench, one or more comparison editors usually open.

Table 19-3 *Preferences: General > Compare/Patch General Tab Settings*

Setting	Description
Open structure compare automatically	Makes visible an additional information area that shows differences in the underlying structure of the resources being compared. This information might not be available for all comparisons. The default is **On**.
Show structure compare in Outline view when possible	Displays the structure compare in the Outline view, whenever it is possible.
Show additional compare information in the status line	Causes the status line to display additional context information about the comparison. The default is **Off**.
Ignore white space	Causes the comparison to ignore differences that are white space characters (for example, spaces and tabs). Also causes differences in line terminators (LF versus CRLF) to be ignored. The default is **Off**.
Automatically save dirty editors before patching	Controls whether any unsaved changes are automatically saved before a patch is applied. The default is **Off**.
Added/ Removed lines	These options control whether a line is counted as added and removed when applying a patch. Both options use regular expressions.
Filtered Members	Specify names, separated by a comma, that are excluded from the **Compare With Each Other** option.

You can change how the text is displayed in the compare option.

Table 19-4 *Preferences: General > Compare/Patch Text Compare Settings*

Setting	Description
Synchronize scrolling between panes in compare viewers	The two comparison viewers lock scroll along with one another to keep identical and corresponding portions of the code in each pane side-by-side. Turn this option off if you don't want the compare viewers to lock scroll.
Initially show ancestor pane	Sometimes you want to compare two versions of a resource with the previous version from which they were both derived. This is called their common ancestor, and it appears in its own comparison pane during a three way compare. Turn this option on if you want the ancestor pane to always appear at the start of a comparison.
Show pseudo conflicts	Displays pseudo conflicts, which occur when two developers make the same change. Turn this option on if you want pseudo conflicts to appear in compare browsers.
Connect ranges with single line	Controls whether differing ranges are visually connected by a single line or a range delimited by two lines.
Highlight individual changes	Controls whether the individual changes inside conflicts are highlighted.
When the end/beginning is reached while navigating an element	Use this option to configure what occurs when the end/beginning is reach while navigating an element.

Content Types

Table 19-5 *Preferences: General > Content Types*

Pane	Description
Content types	The type of content (for example, HTML or XML) that a file contains.
File associations	The file extension that is associated with a content type. For example, `.xml` is associated with a file that contains XML content. To add a file association: 1. Select a content type. 2. Click **Add**. 3. Define a new file type, then click **OK**.

Editors

Table 19-6 *Preferences: General > Editors*

Setting	Description
Size of recently opened files list	The number of files to add to the file menu of recently opened files, which you can easily reopen.
Show multiple editor tabs	Displays tabs for all opened projects. If you deselect this option, only one editor tab displays, and an abbreviated name displays on the tab.
Restore Editor state on startup	Displays the editor in the same state as it was when last closed, as opposed to using default settings.
Prompt to save on close even if still open elsewhere	Saves the file on close even if the same file is open in another editor.
Close editors automatically	Automatically closes the first-opened editor when you open additional editors. This option prevents displaying too many editors and cluttering the workbench.
Number of opened editors before closing	Determines how many editors can be open. For example, if you specify two and then open a third project, the first-opened project automatically closes.
When all editors are dirty or pinned	Prompts you to save unsaved components in the project that is about to automatically close, or to open an additional editor.

- "File Associations" on page 486
- "Hex Editor" on page 487
- "Structured Text Editors" on page 487
- "Text Editors" on page 487

File Associations

Enables you to associate editors (whether they are internally installed in the Designer, or an external application) with file types (extensions) so that you can edit files.

To find out which editor is associated with a file type, select the file type. For example, a `.docgen` file type is associated with the Style editor, but a `.scriptpolicy` file type is associated with the Policy Builder.

To associate an additional editor with a file type:

1. Select the file type.
2. In the **Associated editors** pane, click **Add**.
3. Select an additional editor, then click **OK** twice.

To add a file type:

1. In the **File types** pane, click **Add**.
2. Type the extension (for example, .doc) for the file type, then click **OK**.
3. In the **Associated editors** pane, click **Add**.
4. Select an editor for that file type, then click **OK** twice.

Hex Editor

Enables you to configure Designer's hex editor environment, including font, font style, and colors. You can also associate, or disassociate, the hex editor from Designer's registered file extensions, and enable hex editor logging.

Structured Text Editors

For information about structured text editors, refer to the Eclipse documentation (http://help.eclipse.org/helios/index.jsp).

Text Editors

Table 19-7 *Preferences: General > Text Editors*

Setting	Description
Undo history size	Determines the size of the undo history. The default is 200 changes.
Displayed tab width	Specifies the number of characters or spaces in a tab character. The default is 4. The maximum is 16.
Insert spaces for tabs	Inserts the number of spaces specified in **Displayed Tab Width**, instead of a tab character, when you press the tab key in the text editor.
Highlight current line	Highlights the current line.
Show print margin	Displays the print margin on the right side of the text document. A vertical line identifies the margin.
Show line numbers	Numbers each line in the editor.
Show range indicator	Displays a range indicator.
Show whitespace characters	Displays white space characters so you can see them in the text editor.
Enable drag and drop of text	Allows you to drag and drop text within the text editor.
Warn before editing a derived file	Notifies you if you attempt to edit a file generated or maintained by the system. Your changes might be overwritten.

Setting	Description
Smart caret positioning at the line start and end	Enables the Home and End commands to move to the first and last non-white-space character on a line.
Show affordance in hover on how to make it sticky	Enables the hover over text to grab the text and place it in the clipboard.
Appearance color options	Lets you configure the display settings for the text editor. Select a particular appearance characteristic from the list to view and change the display settings for that characteristic.

For additional information about text editors, see the Eclipse documentation.

Keys

Enables you to view a table of all of the keyboard mappings, change those mappings, and add new mappings.

Network Connections

Enables you to configure a manual proxy configuration if you use a proxy server to access the Internet. For example, if you have added a custom URL for packages that require authentication, you must enter that information here so auto updates of packages works.

The three options are:

Table 19-8 *Preferences: General Settings > Network Connections*

Settings	Description
System proxy configuration (if available)	Specifies that the system proxy settings are used to access the Internet. If the settings can't be retrieved, no proxy should be used.
Direction connection to the Internet	Select this option if no authentication information is required. This is the default option.
Manual proxy configuration	Specify that a proxy server is required to access the Internet. Select **Enable proxy authentication** if you have specified a URL that requires authentication. For example, if you have added a URL to download custom packages, you must specify the username and password here.

Perspectives

Table 19-9 *Preferences: General > Perspectives*

Setting	Description
Open a new perspective	**In the same window**: Places a new icon in the Perspective Switcher, so that you can toggle between perspectives in the same window. **In a new window**: Opens a new perspective in a different window. You can toggle between perspective windows by selecting icons on the taskbar.
Open a new view	**Within the perspective**: Opens the view so that it is contiguous to the Modeler. **As fast view**: Opens the view and places a Fast View in the bottom left corner of the perspective.
Open the associated perspective when creating a new project	Determines how and when you switch to an associated editor when you open a perspective.
Available perspectives	Designer is the default perspective. Other available perspectives are Eclipse Debug and Resource.

Startup and Shutdown

Table 19-10 *Preferences: General > Startup and Shutdown*

Setting	Description
Prompt for workspace on startup	Prompts you for a workspace folder. You can have multiple workspace folders and can specify a folder on startup.
Refresh workspace on startup	Synchronizes the workspace with resources (for example, `myfile.xml`) on disk.
Confirm exit when closing last window	Displays an **Exit Designer**? prompt when you exit Designer.
Plug-ins activated on startup	Lists plug-ins that are automatically loaded and registered.

Web Browser

Table 19-11 *Preferences: General > Web Browser*

Setting	Description
Use internal Web browser	Enables you to use an internal Web browser.
Use external Web browser	Enables you to add and use an external browser (for example, Netscape). If you enable this option, you must also enable **Use External Browser** in the **Help** section (also found in Preferences).
External Web browsers	Lists browsers. To add a browser: 1. Click **New**. 2. Name the new browser. 3. Scroll to and select an executable (for example, `netscp6.exe`). 4. Specify a parameter, then click **OK**.

Welcome

Table 19-12 *Preferences: General > Welcome*

Setting	Description
Home: Home Page Theme	Enables you to select the theme that appears when you click **Help** > **Welcome**.
Home: Root Pages	Adds tabs (for example, **Overview**) on the Welcome properties page. You add functionality by customizing these tabs.

For information about the Overview and **What's New** tabs, refer to the Eclipse documentation.

Help

Table 19-13 *Preferences: Help*

Setting	Description
Specify how help information is displayed: Use external browser	If an embedded Web browser is supported on your system, the Help view uses that browser to display help contents. To force help to use an external browser, enable this option. Specify an external browser in Preferences: **General** > **Web Browser**.
Open window context help	Determines whether the window context help opens in a dynamic Help view or in a pop-up window.
Open dialog context help	Determines whether the dialog box context help opens in a dynamic help section of the Help view or in a pop-up window.
Open help view documents	Determines whether the documents selected in the Help view open in place or in the editor area.

Content

Designer lets you include external information in the help system.

Table 19-14 *Preferences: Help > Content Settings*

Setting	Description
Include help content from a remote infocenter	Enables including external information in the help system.
Location	Specifies the hostname, path, and port to the external information.

NetIQ

The following Preferences categories appear as NetIQ sub-pages:

- “Designer” on page 492
- “OpenText Identity Manager” on page 495
- “Package Manager” on page 511
- “Provisioning” on page 514

Designer

The following preferences categories appear as Designer sub-pages:

- "DS Trace" on page 492
- "Editor" on page 492
- "JavaScript Validation" on page 493
- "LDAP Connection" on page 493
- "Language" on page 493
- "Project Checker" on page 494
- "Schema" on page 494
- "Trace" on page 494
- "Version Control" on page 495

DS Trace

Let us configure DS Trace settings.

Table 19-15 *DS Trace Preferences*

Setting	Description
Live DS Trace Display	Specifies the size of the DS Trace window buffer, in lines (or entries). When the number of DS Trace entries exceeds the Window Size, DS Trace drops the oldest entry for each new entry it captures.
Auto-scroll display	Enables auto-scrolling of the live DS Trace window so that the latest log entries are always on screen. When this option is deselected, you must manually scroll down the list of log entries.

Editor

The **Editor** option allows you to set the tab limit. That is, you can set a limit on how many tabs can be opened while working on Designer. Though Designer allows you to open tabs more than the set limit, this may affect the Designer performance.

Table 19-16 Preferences > NetIQ > Designer > Editor

Setting	Description
Set the maximum tab limit	Specify the maximum tab limit. By default, the value is set to 20. **NOTE:** This is a mandatory field. You must specify any number except zero.
Display the warning message on maximum tab limit	Selecting this option displays a warning message if you open tabs more than the maximum set limit. **NOTE:** If you open tabs more than the set limit, a warning message appears. You can continue to open tabs more than the set limit however, this would impact the Designer performance. It is recommended that you close the unused tabs.

JavaScript Validation

Designer automatically validates the JavaScript as it is typed into the UI. By default, it is enabled.

LDAP Connection

Lets you configure the LDAP connection settings for authenticating to the Identity Vault while importing, deploying, or comparing Designer objects with the Identity Vault.

Table 19-17 LDAP Connection Preferences

Setting	Description
Certificate Password	Specifies the password to access the server certificate in `LDAPServerCerts` keystore in Designer 's configuration directory (`/opt/netiq/tools/Designer/configuration` or `C:\netiq\idm\apps\Designer\configuration`). Designer uses this certificate to connect to the Identity Vault.
Prompt to show certificate import dialog	Instructs Designer to prompt you to accept the certificate each time you authenticate to the Identity Vault.
Remember user selection for importing certificate	Allows you to turn off this prompt and instructs Designer to remember this setting for future authentication.

Language

When you installed Designer, you selected a language to display Designer's UI. This setting enables you to change the language.

1. Navigate to **Preferences > NetIQ > Designer > Language**.
2. Select a language, then click **OK**.

You must restart Designer for the language change to take effect.

3 Restart Designer.

NOTE: Restore Defaults reads the `config.ini` file, detects the previous language setting, and then defaults to that setting. When the changed property is written back to the `.ini` file, all comments are removed from the file. To preserve these comments, Designer copies the original `config.ini` to `config.ini.bak` and uses the backup to determine the default setting.

Project Checker

Lets you configure the Project Checker.

Table 19-18 *Preferences: NetIQ > Designer > Project Checker*

Setting	Description
Limit Visible Items to	Allows you to limit the number of items displayed in the Project Checker. The default value is 100.
Prompt me to save the editor before running Project Checker	Allows you to receive a prompt asking you to save your project before running the Project Checker. By default, this is enabled.

Schema

Allows you to manage the Identity Vault and managed system's schema.

Table 19-19 *Preferences: NetIQ > Designer > Schema*

Setting	Description
Warn when LDAP names are different from eDirectory names during .ldif import/export	Allows you to turn off this warning prompt, which appears during the import or export of the schema.
Warn when exporting base classes to .ldif	Allows you to turn off this warning prompt, which appears during the export of the schema.
Show the information message for the Manage Application Schema context menu	Allows you to turn off the information message that appears when managing the application schema.

Trace

The Trace view is useful in the following situations:

- To trace internal errors and messages, so that you can find out why something might not work as expected.
- To provide information for OpenText Support, engineers, or other consulting resources.

All Designer-specific trace messages go to the Trace view if this view is open. Otherwise, no trace messages are sent.

Warnings and error messages are sent to the `.log` file, found in the run-time workspace `metadata` directory. Use the Error view to view this information.

Table 19-20 *Parameters: NetIQ > Designer > Trace*

Setting	Description
Enable tracing	Writes events to the Trace view. By default, tracing is off. To increase performance, disable tracing when you don't need it.
Include stack traces	Provides separate traces. Dumps the entire stack where an internal exception occurs, so that you can see in the code where the internal exception is failing.
Include XML processor traces	Provides separate traces that detail all of the processing of XML documents. This trace can become quite verbose.
Show plug-In names in the trace	In the Trace view, displays names of plug-ins where tracing has occurred. This is useful if you are tracing more than one plug-in.
Show view when tracing	Automatically brings up the Trace view if a trace is trying to be logged. By default, this setting is On.
Trace buffer size	Increases the buffer to show more characters. As the buffer increases in size gets higher, performance might degrade, depending on your system.
Plug-Ins to Trace	Lists all Designer plug-ins (in their simple name form). Select plug-ins that you want to trace.
Select All	Enables tracing in all Designer plug-ins.
Deselect All	Disables tracing in all Designer plug-ins.

Version Control

This setting determines how often Version Control polls the SVN server for updates. The polling interval is in minutes.

OpenText Identity Manager

The following preferences categories appear as OpenText Identity Manager sub-pages:

- "OpenText Identity Manager" on page 496
- "Configuration" on page 498
- "Document Generation" on page 501

- “Entitlements” on page 502
- “Import/Deploy” on page 502
- “Modeler” on page 505
- “NAT Mapping” on page 507
- “Policy Builder” on page 509
- “Simulation” on page 510
- “iManager” on page 510

OpenText Identity Manager

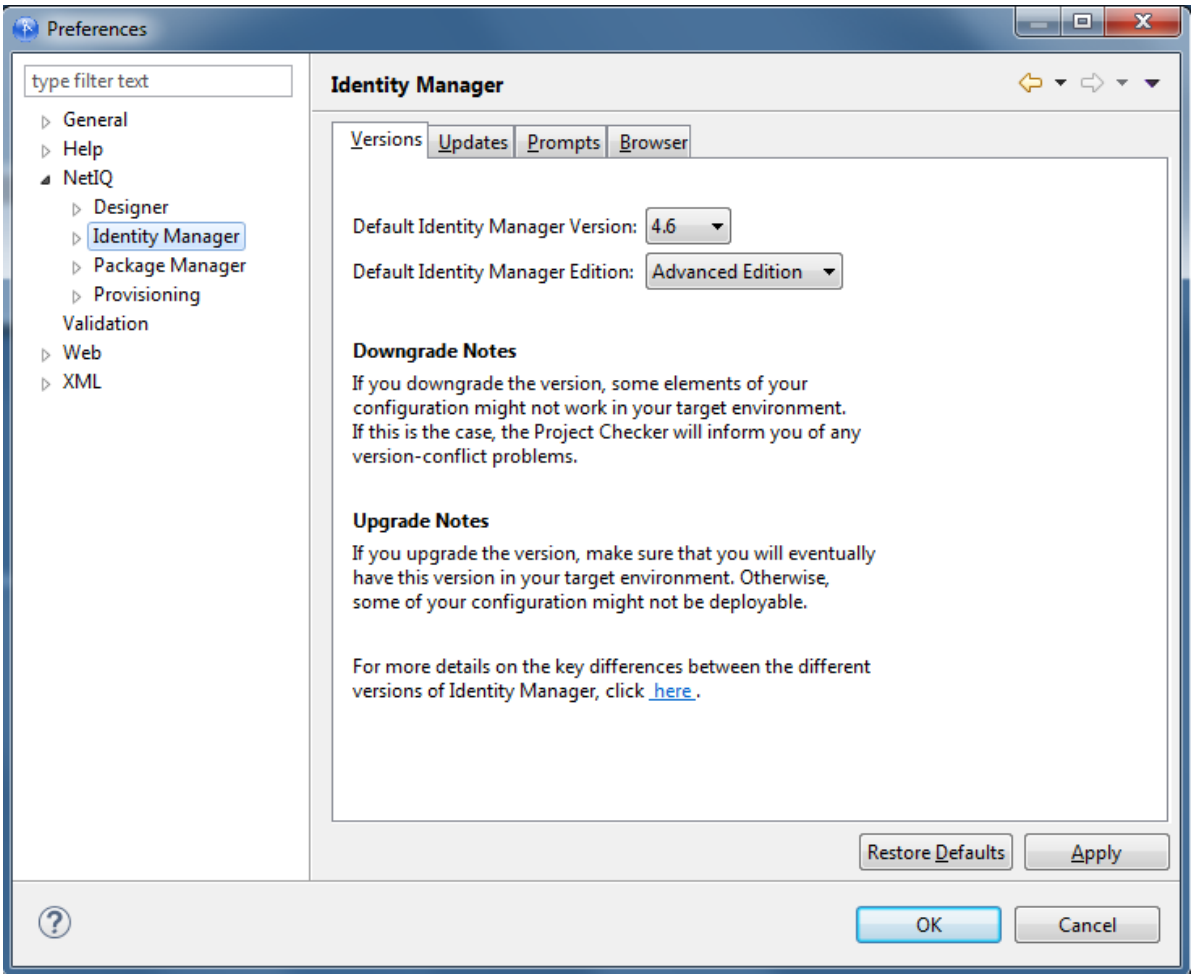
The OpenText Identity Manager option contains multiple tabs:

- “Versions” on page 496
- “Updates” on page 497
- “Prompts” on page 498
- “Browser” on page 498

Versions

Specifies the OpenText Identity Manager version running on a server.

Figure 19-2 *Preferences: NetIQ > Identity Manager > Versions*



Updates

Table 19-21 *Preferences: NetIQ > Identity Manager > Updates*

Settings	Description
Do not check for updates	Prevents Designer from checking for updates on startup. Hides the Designer Updates dialog box.
Prompt to check for updates on startup	Displays a prompt each time you run Designer. You can disable this prompt.
Automatically check for updates on startup	Always checks for updates. If you disable the prompts that appear on startup, select this option.
Notify me when no updates are available	Displays a No New Updates message when you select to check for updates.

Prompts

Table 19-22 *Preferences: NetIQ > Identity Manager > Prompts*

Setting	Description
Warn when downgrading server versions	Prompts you when you select an earlier server version for a project. If you downgrade, some elements of your configuration might not work in your target environment.
Warn when upgrading server versions	Prompts you when you select a later server version for a project. If you upgrade, some of your configuration might not be deployable unless you have this later server version in your environment.
Warn when another editor has updated files in the same project space	Warns you that your project might be erased from your workspace. The prompt occurs when overwriting a file in the file system for notification templates and policies.
Warn when deleting items from the outline view	Confirms that you want to delete the selected items.

Browser

You can use Designer to open a Web browser. After you enter the URL, Designer stores it. To change the URL, type a new one in **Preferences**, then click **OK**.

Configuration

- “General” on page 498
- “eDir-to-eDir SSL/TLS” on page 499
- “Prompts” on page 501

Each driver has a startup parameter. If it is disabled, the driver never starts until you change the setting. By default, OpenText Identity Manager drivers are disabled when you create them in the Modeler or start Designer. You must start them manually.

For more information, see “Configuring Driver Sets” on page 76.

General

These general settings specify how drivers start up and how their global configuration values (GCVs) act on specified target servers. The default state uses **Disabled** and **Merge GCVs**.

Table 19-23 Preferences: NetIQ > Identity Manager > Configuration > General Tab Settings

Setting	Description
Auto-Start	The driver automatically starts after you create it or whenever you start or load Designer.
Manual	You must start the driver manually.
Disabled	The driver never starts.
Merge GCVs on the target server during copy	Copies the GCVs from one driver/driver set to multiple targets of the same type. For example, you might configure GCVs on one driver and then copy them to multiple drivers. You also have the option of overwriting the target GCVs or merging your source GCVs with the existing target driver GCVs, if they exist.
Overwrite GCVs on the target server during copy	Overwrites existing GCVs when they are copied to the server.
Set ECVs on all the Associated Servers during package installation	Sets ECVs for all the associated servers during package installation. If you disable the setting, the ECVs will only be set to the selected server.

eDir-to-eDir SSL/TLS

This setting configures how two OpenText eDirectory drivers communicate with each other over a secure channel. For more information, see "Configuring TLS for eDir-to-eDir Drivers" on page 435.

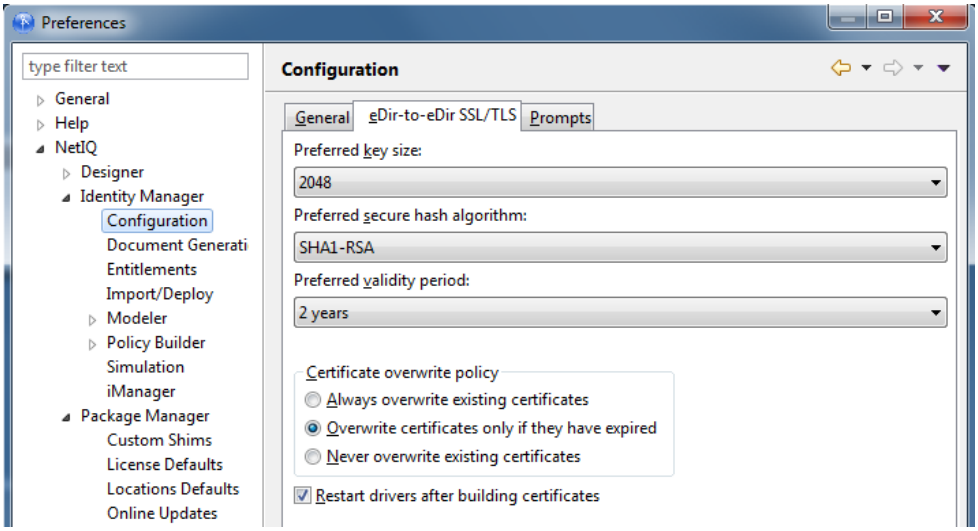
Table 19-24 Preferences: NetIQ > Identity Manager > Configuration > eDir-to-eDir SSL/TLS Tab Settings

Setting	Description
Preferred key size	Specifies the preferred key size that is generated when drivers are encrypted and stored in OpenText eDirectory: 256, 384, 512, 768, 1024, or 2048 bytes.
Preferred secure hash algorithm	Specifies the preferred hash algorithm to use when encrypting drivers: SHA1-RSA, MD2-RSA, MD5-RSA, SHA256-ECDSA, or SHA384-ECDSA. ◆ For 256 key size, Designer provides SHA256-ECDSA algorithm. ◆ For 384 key size, Designer provides SHA384-ECDSA algorithm. SHA256-ECDSA and SHA384-ECDSA are Suite B-compliant algorithms.
Preferred validity period	Specifies the validity period for a driver certificate, ranging from 6 months to 10 years.

Setting	Description
Always overwrite existing certificates	Specifies that existing driver certificates are overwritten with each deployment. If you select this option, Designer deletes existing certificates and creates new ones. The new certificates are then good for another two years (assuming the default value is two years, as defined in the Preferred Validity Period field.) If you select Live > Create eDir-to-eDir Certificates, Designer deletes old certificates and creates new ones.
Overwrite certificates only if they have expired	Specifies that only expired driver certificates are overwritten with each deployment. This is the default setting. The default expiration length is two years. If a certificate expires, SSL/TLS stops working. If a certificate is expired, Designer deletes it and creates a new one.
Never overwrite existing certificates	Never overwrites driver certificates.
Restart drivers after building certificates	Restarts drivers after certificates have been updated or created.

When you create certificates, Designer reads the preferences, including Preferred Key Size, Preferred Secure Hash Algorithm, and Preferred Validity Period. These options are also available through Secure Connection Settings > Advanced TLS Configuration.

Figure 19-3 *The Advanced TLS Configuration Dialog Box*



NOTE: Designer reads these preferences after you first set them. If you subsequently change the preferences by using the driver's configuration page, those changes override the settings in Preferences.

After you change default settings and click OK, that configuration information is recorded. When you deploy the driver, Designer creates the certificates, or deletes and creates new certificates with a new time stamp.

Prompts

These settings specify how users are prompted to manage driver certificates on the target server. All are selected in the default state.

Table 19-25 *Preferences: NetIQ > Identity Manager > Configuration > Prompts Tab Settings*

Setting	Description
Prompt to replace existing certificates	Prompts the user to provide new certificates.
Prompt to merge/overwrite GCVs on target server during copy	Prompts the user to merge or overwrite when copying GCVs to the target server.
Prompt to create certificates after configuration	Prompts the user to create certificates after configuring a secure connection.
Prompt to overwrite existing settings and policies from the Driver Configuration Wizard	In the Driver Configuration Wizard, prompts the user whether to reset (overwrite) all driver settings and policies.
Prompt when policy operations affect multiple policy sets	Turns on and turns off a warning dialog box associated with policy operations. The dialog box appears when you move policies in a pre-3.5 environment and the move operation affects multiple policy sets.
Prompt for server selection on live driver actions	Any time you perform a live action on a driver (such as starting or stopping the driver) it prompts you to specify the server associated with the driver.
Prompts for errors when validating XML DTD for all Policy Editors	Designer validates the policies you create against the OpenText Identity Manager DTDs. This helps you verify that the policies you create are valid.

Document Generation

The Document Generator comes with the following settings:

Table 19-26 *Preferences: NetIQ > Identity Manager > Document Generation*

Setting	Description
Automatically open the rendered file after document generation.	If you have a PDF reader installed on your workstation, the rendered file automatically opens in the reader. If you have enabled the RTF format and have a TRTF reader installed, the rendered file automatically opens in the reader. The default is On.
Show warning dialog box when the style is an older version.	Displays a warning when generating documents on out-of-version styles. The default is On.

Setting	Description
Warn me before overwriting existing file during document generation	Displays a warning when overwriting previously generated files.
Enable RTF support	Allows you to save documents to RTF format. The default is Off.
Output XML source files	Generates XML files as part of the document generation process.
Document applications and drivers related to other selected items.	With this option selected, parent objects and direct child objects are included to give context to the document. Deselecting this option excludes direct children of the selected item. The default is On.
Document Language	Allows you to select a language other then English in which to generate documents. Languages include Chinese Simplified, Chinese Traditional, Dutch, English, French, German, Italian, Japanese, Portuguese Brazil, and Spanish. The default is English.
Font settings	Allows you to select the font you want to use for document generation. This selection adds double-byte font support. The default is the Arial font.

Entitlements

Controls whether or how often you receive a prompt whenever you add the DirXML-EntitlementRef attribute to a driver filter. The default is **Prompt me**, but because this attribute is added only if it does not already exist on the driver filter, you can select **Always add it** to not see the pop-up window.

You can also never add the attribute. However, the DirXML-EntitlementRef attribute is added only if it does not already exist in the driver filter. If the attribute already exists, the options have no effect.

Import/Deploy

The Import/Deploy preferences window contains three tabs: **Behaviors**, **Prompts**, **Trace** and **General**. The following tables describe their options.

- "Behaviors" on page 502
- "Prompts" on page 504
- "Trace" on page 504
- "General" on page 504

Behaviors

There are multiple sections in the **Behaviors** tab.

Table 19-27 *Preferences: NetIQ > Identity Manager > Import/Deploy > Behaviors (Import Settings)*

Setting	Description
Perform prompt checking when running a driver configuration file	Displays the **Do you wish to perform all mandatory and required prompt checking when running this Driver configuration file?** prompt. If you select **Yes** to the prompt, you must then enter information in required fields while configuring the driver. If you select **No**, you temporarily disable this setting and can skip required fields.
Include application schema when importing drivers	Imports the OpenText eDirectory application schema when you select this option. You might not want to import all the associated data. The default is Off. See "Importing a Schema" on page 312.

Table 19-28 *Preferences: NetIQ > Identity Manager > Import/Deploy> Behaviors (Deploy Settings)*

Setting	Description
Replace driver set/server associations when deploying a driver set	If you want driver set and server associations when deploying, select this option. The default is Off.
Always deploy both drivers of an eDir-to-eDir connection	With this option selected, you are prompted to deploy both sides on the connection. With both drivers deployed, Deploy is integrated with the creation of eDir-to-eDir certificates, if the certificates are created in Designer. Deploy adheres to the settings set in **Preferences > Designer for IDM > Configuration > eDir-to-eDir.** The default is On. This is the recommended setting.
Restart running drivers after deploying the driver	Restarts the driver after it is deployed. The default is On.

Table 19-29 *Preferences: NetIQ > Identity Manager > Import/Deploy > Behaviors (Summary Dialog)*

Setting	Description
Show the summary dialog prior to performing an import	Allows you to view what's being imported in a summary screen. The default is On.
Show the summary dialog prior to performing a deployment	Allows you to view what's being deployed in a summary screen. The default is On.
Filter passwords out of summary and compare dialogs	Select this box if you want to filter passwords out of summary and compare dialog boxes.

Table 19-30 *Preferences: NetIQ > Identity Manager > Import/Deploy >Behaviors (Export Settings)*

Setting	Description
Copy cross driver policy references into exported configuration files	Selected by default, this option saves you the trouble of manually inputting cross-driver policy references.

Prompts

Table 19-31 *Preferences: NetIQ > Identity Manager > Import/Deploy > Prompts Tab Settings*

Setting	Description
Show dialog to export cross driver policy references to configuration files	Selected by default. If you do not want to see a dialog box about these references, deselect the option.
Show a warning dialog when overwriting a driver set/server association	Warns that the driver set being deployed has a different server association than the server that you are about to deploy to. The association in the deployed driver set overwrites the existing server association.
Show the dialog box to deploy both drivers of an eDir-to-eDir connection	This is the default, and it is also the recommended setting. With this option selected, you are prompted to deploy both sides of the connection.
Show the dialog box to restart drivers after a deployment	Selected by default. If you do not want to see a dialog box about these references, deselect the option.

Trace

Table 19-32 *Preferences: NetIQ > Identity Manager > Import/Deploy > Trace Tab Settings*

Setting	Description
Trace import and deploy event information	Deselected by default. If you need to troubleshoot an import or a deploy, select this option, then open the Trace view to inspect the import or deploy.
Generate debug messages for the Driver configuration prompt dialog box	Deselected by default. If you need to generate debug messages, select this option.
Show verbose debug messages	Deselected by default. If you need to generate verbose debug messages, select this option.
Time import and deploy operations	Deselected by default. If you need to time how long it takes to import or deploy an object, select this option.

General

OpenText Identity Manager allows you to configure the limit for displaying results while browsing the Identity Vault. For example, if you are defining a security equivalence for a driver, the Browse Identity Vault window displays the number of users or groups based on the configured limit.

To configure the display result limit, browse to **Windows > Preferences > NetIQ > Identity Manager > Import/Deploy > General** tab and set the value in **Identity Vault browser max results to return** field. The default value is 7000.

Modeler

The Modeler preferences window contains seven tabs: **Behaviors**, **Display**, **Guidance**, **Layouts**, **Pages**, **Prompts**, and **Themes**. The following tables describe their options.

Additionally, the following preferences categories appear as Modeler sub-pages:

- "Dataflow Page" on page 507
- "Palette Page" on page 507

Table 19-33 *Preferences: NetIQ > Identity Manager > Modeler > Behaviors Tab Settings*

Setting	Description
Auto-create servers when connecting a driver to a different driver set	Automatically creates a server for a driver set when you connect a driver to a different driver set.
Launch the driver Properties dialog box	Launches the driver's Properties page.
Show the driver's Policy Flow view	Displays the driver's Policy Flow diagram in the Outline view.

Table 19-34 *Preferences: NetIQ > Identity Manager > Modeler > Display Tab Settings*

Setting	Description
Show labels by Applications and Identity Vaults (Architect. mode)	Shows labels below applications (in both modes) and above Identity Vaults (Architect mode only).
Show driver icons in Developer mode	Displays a driver icon on the line that represents a driver in the Modeler.
Show password icons in Developer mode	Displays a password sync icon below a driver icon in the Modeler.
Auto-expand Identity Vaults to fit contents	Causes Identity Vaults to expand to accommodate objects that you place in them.
Auto-shrink Identity Vaults to fit contents	Causes Identity Vaults to shrink when you remove objects from them.
Auto-size Identity Vaults to fit their titles	Enables vaults to expand horizontally, to accommodate long titles. Otherwise, the titles concatenate after approximately 20 characters.
Grid Width	Increases or decreases cells in the Modeler's grid. To access the grid, select the Modeler, then click **View** > **Grid**.

Table 19-35 *Preferences: NetIQ > Identity Manager > Modeler > Guidance Tab Settings*

Setting	Description
If an Identity Vault doesn't already exist, one will be created when you drop the application	Creates an Identity Vault when you drag or drop an application from the palette into the Modeler.
eDir-to-eDir connection tip, when you've connected the same eDir app to two driver sets	Prompts you to connect a line directly between the end driver sets when you set up an eDir-to-eDir relationship.
Setting dataflows in architect mode will default all policy and schema settings	Sets policy and schema settings to defaults when you set data flows in architect mode. To edit the settings, use the Developer mode.
Saving Dataflow to disk will first force a project save	Requires you to save a project before you can save a dataflow to disk.

Table 19-36 *Preferences: NetIQ > Identity Manager > Modeler > Layouts Tab Settings*

Setting	Description
Default Layout for Applications on Import	Specifies the default layout for application objects when you import a project into Designer.

To arrange an existing project in a particular layout:

1. In the Modeler, right-click a driver set.
2. Select **Arrange Applications**.
3. Select a layout.

Table 19-37 *Preferences: NetIQ > Identity Manager > Modeler > Pages Tab Settings*

Setting	Description
Check the additional Modeler pages you want visible	Determines whether the Architect, Dataflow, and Table pages display as tabs at the bottom of the Modeler. The Developer mode is always enabled.

Table 19-38 *Preferences: NetIQ > Identity Manager > Modeler > Prompts Tab Settings*

Setting	Description
Show the Driver Config Wizard at connection time	Launches the Driver Configuration Wizard when you drag or drop an application in the Modeler.
Confirm when a driver is being deleted	Provides a Yes/No prompt for you to choose whether you want to delete the driver and its policies.

Table 19-39 Preferences: NetIQ > Identity Manager > Modeler > Themes Tab Settings

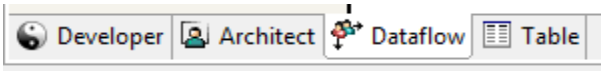
Setting	Description
Developer	Specifies the theme for Developer mode. Themes define the colors used for background, text, line, domain group background, and domain group title in the Modeler.
Architect	Specifies the theme for Architect mode. Themes define the colors used for background, text, line, domain group background, and domain group title in the Modeler.

Dataflow Page

Specifies the number of columns per page that the Dataflow editor saves in the HTML reports.

To view or use the Dataflow editor, select the Dataflow tab in the Modeler.

Figure 19-4 The Dataflow Tab



Palette Page

The Palette page includes the following settings:

Table 19-40 Preferences: NetIQ > Identity Manager > Modeler > Palette

Setting	Description
Arrange applications in folders	Displays folders (for example, Database) in the palette and places applications in appropriate folders.
Arrange applications in an alphabetical list	Places all applications into one folder in the palette, and lists the applications alphabetically.

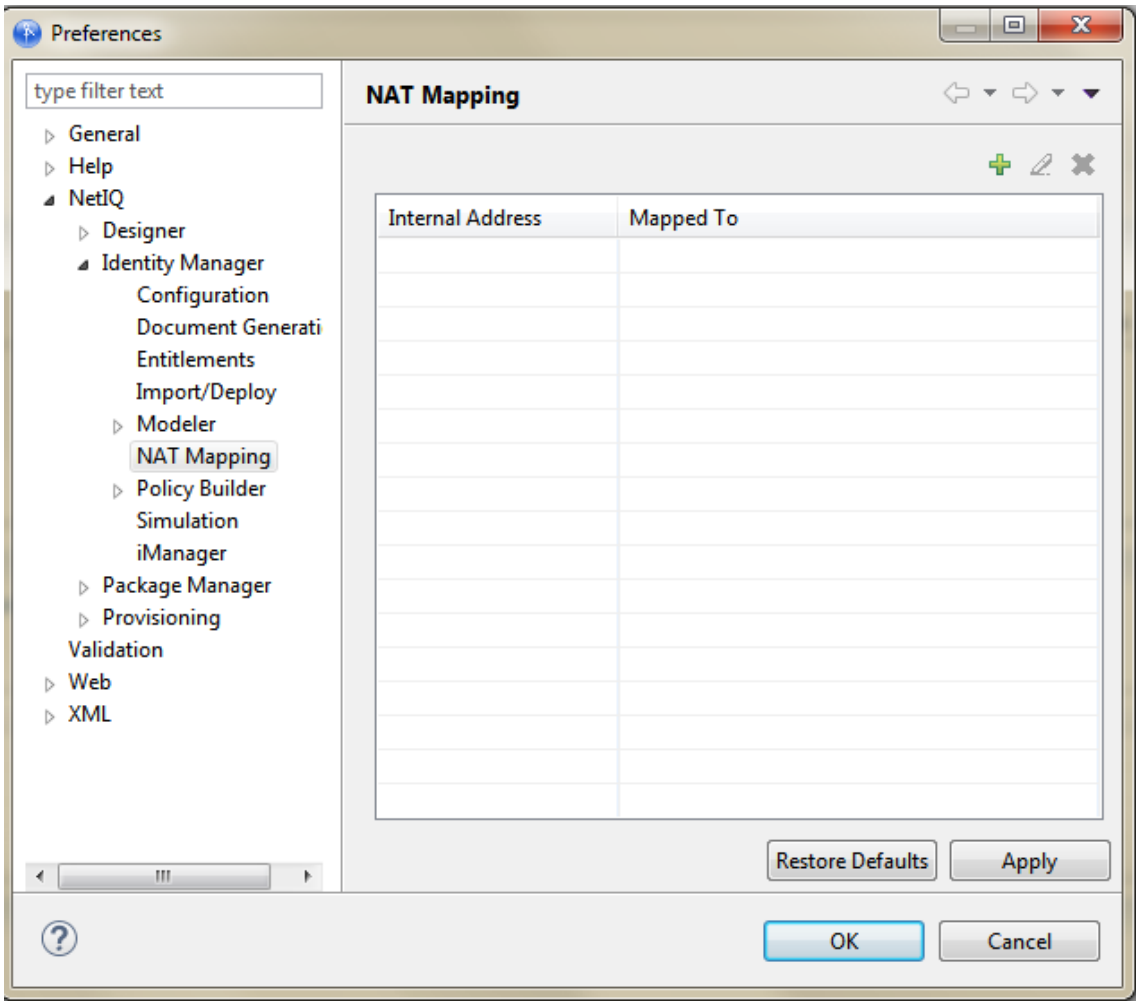
NAT Mapping

If the server associated with your driver set is located in a private network, Designer provides an option to map the server's local IP address to an external IP address in a NAT (Network Address Translation) environment. Designer (LDAP) allows a one-to-one mapping between the local and global IP addresses.

To import or deploy server specific values, Designer (LDAP) creates a separate connection with each server associated with the driver set. When you add a new server to the driver set, Designer (LDAP) obtains the IP address of the server stored in the Identity Vault before connecting to the server. If the server resides in a private network, the connection succeeds only when the IP address of the server is mapped to an external IP address. After NAT mapping is configured for the server, Designer (LDAP) uses the mapped external IP address of the server to establish a connection with the server.

To add a new mapping entry, go to Preferences and click **Window** > **Preferences** > **NetIQ** > **Identity Manager** > **NAT Mapping**.

1. Click  icon.



2. In the window that displays, specify the internal and external IP addresses that you want to map to each other.
3. Click **OK**.
4. Click **Apply** to exit the Preferences page.

Alternatively, add the external IP address in a text file and place it in `/opt/netiq/tools/Designer/configuration` or `C:\netiq\idm\apps\Designer\configuration` directory. For the change to take effect, restart Designer.

Port Mapping

NAT mapping allows you to map only the internal address to the external address. With Port Mapping, you can now add the port number for the internal and external addresses. You can specify the port numbers in the address fields.

If the port number is unspecified, the default port number will be based on the value that you specified during the Identity Vault Configuration. For example, the port number will be defaulted to 636 for secure connections and 389 for non-secure connections.

Policy Builder

The Policy Builder preferences page includes the following settings:

Additionally, the following preferences categories appear as Policy Builder sub-pages:

Table 19-41 *Preferences: NetIQ > Identity Manager > Policy Builder*

Setting	Description
Localize actions, conditions and tokens	Translates the names of policy actions, conditions and tokens into the selected Designer language. When this option is not selected, policy actions, conditions and tokens display in English.
Include project name in title	Includes project name in the title.
Expand all rules when the Policy Builder is loaded	Automatically expands rules in the Rules pane when you open the Policy Builder.
Show version/author/last changed information	Adds additional fields in the Rule Inline editor (available when you double-click a rule.) Designer adds the information from these fields to the policy.

- "Policy Description" on page 509

Policy Description

The Policy Description preferences page includes the following settings:

Table 19-42 *Preferences: NetIQ > Identity Manager > Policy Builder > Policy Description*

Setting	Description
Expand the Policy Description field	Automatically expands the **Policy Description** field. You can hide the field by selecting the check box.
Number of rows of text to display	Determines how many rows to display in the **Policy Description** field. The default is 10.
Policy Description position on the page	Places the **Policy Description** field above or below the Rules pane.

Simulation

The Simulation preferences page includes the following settings:

Table 19-43 *Preferences: NetIQ > Identity Manager > Simulation*

Setting	Description
Directories: Java Extensions	Enables you to simulate policies that contain references to external Java extensions. Specify the `.jar` file or the directory where the `.jar` file is located to add it to the class path. You can specify multiple Java extensions.
Referenced Directories	A reference directory table and a new configuration option to specify the current working directory have been added in the Simulation preferences page.You can add directories through this table when they need to be included in the Simulator's classpath. The configuration or reference files in the directory are available at runtime while simulating the policy.
Options: Clear the policy simulation log file prior to performing a simulation	Automatically clears the log file. If you don't enable this setting, Designer displays a **Clear Log** icon that you can use. If you do many simulations in succession, you might want to disable this option. The log file then captures and displays the events of all the tests, until you click **Clear Log**.
Options: Show the information prompt when a query is generated	Displays a prompt when the Simulator generates a query. It simulates what the engine would do when a query is required to process the policy.
Options: Notify user when converting the Input Document schema	Notifies a user when the Policy Simulator must convert the Application schema to the ID Vault schema, or vice versa. This is typically necessary when changing the input document's simulation point.

iManager

The iManager preferences page includes the following settings:

Table 19-44 *Preferences: NetIQ > Identity Manager > iManager*

Setting	Description
iManager URL	The IP address and port for the iManager server.
Show iManager Information Dialog	Prompts you for the URL to the iManager server after you select **Tools > iManager**. If the URL is missing or incorrect, iManager is unable to launch.

Package Manager

The following options allow you to manage packages in Designer. You access the preferences page through **Windows > Preferences > NetIQ > Package Manager**.

- "Auto Imports" on page 511
- "Custom Shims" on page 511
- "License Defaults" on page 512
- "Locations Defaults" on page 512
- "Online Updates" on page 513
- "Package Based Policies" on page 514
- "Vendor Information" on page 514

Auto Imports

This setting allows you to change how Designer imports package updates into the package catalog. When there are updates to packages that have not been imported into the package catalog, select how you want Designer to handle these updates.

Table 19-45 *Preferences: NetIQ > Package Manager > Auto Imports*

Setting	Description
Do not import packages when a project opens	Designer does not prompt you to import updated packages into the package catalog. If there are package updates that need to be imported, you must manually import these packages before they can be installed. For more information, see "Importing Packages into the Package Catalog" on page 180.
Prompts to import packages when a project opens	If there are package updates, every time you open the project, you are prompted to import the package updates into the package catalog.
Automatically import packages when a project opens	If there are package updates, every time you open the project, Designer automatically imports the package updates into the package catalog.

Custom Shims

Allows a developer to specify information about a custom driver shim. The information is used as a template so that a developer does not need to specify this information repeatedly when creating a package.

Table 19-46 Preferences: NetIQ > Package Manager > Custom Shims

Setting	Description
Display Name	Displays the driver name and version in the driver manifest. This name can change with each release of the driver.
Shim ID	Associates the driver with the shim file in the driver manifest. This ID never changes.
Driver Palette ID	This ID associates the driver shim with a certain types of drivers. This allows you to group packages together. For example, if your driver palette ID associates your custom driver with the JDBC driver, your packages are available for installation if the customer has a JDBC base package installed.

To add a custom shim:

1. Click the **Add shim type** icon .
2. Specify the display name for the driver shim.
3. Specify the shim ID for the driver shim.
4. Specify the driver palette ID for the drivers you want this custom shim to be associated with.
5. Click **Apply**.

License Defaults

If you have a license for packages you are developing, you can specify that information in this preference page, so that each time you create a new package you don't need to specify that information again.

To add a license:

1. Click **Browse**, then browse to and select your license file.
2. Click **Apply**.

Locations Defaults

This option allows you to specify your package development directories so that you don't need to specify this information each time you create a new package.

Table 19-47 Preferences: NetIQ > Identity Manager > Package Manager > Location Defaults

Setting	Description
Build Directory	This directory is where you build packages.
Import Directory	This directory contains all imported packages.
Localization Directory	This directory contains all of the packages that are localized.
Publish Directory	This directory contains all packages ready to publish.

Online Updates

The following settings configure how packages are updated online:

Table 19-48 *Preferences: NetIQ > Package Manager > Online Updates*

Setting	Description
Do not check for updates	Designer does not automatically check for updates. With this option selected, you need to manually check for updates by clicking **Help** > **Check for Package Updates** in Designer's toolbar.
Prompt to check for updates on startup	Designer prompts you to check for package updates when it starts.
Automatically check for updates on startup	Designer checks for any package updates when it starts. **NOTE:** This options fails if a custom site requires authentication and the authentication information has not been added in DesignerDesigner. You add the authentication information into **Window** > **Preferences** > **General Settings** > **Network Connections**. For more information, see "Network Connections" on page 488.
Notify me when no updates are available	If there are no package updates, Designer returns a message stating that no updates are available.
Package Update URLs	Lists the URLs where Designer checks for package updates. Partners can add their own URLs for custom packages. For more information see, "Releasing and Publishing Packages" on page 240.
Add URL	Allows you to add the vendor's name and URL for publishing custom packages. For more information, see "Releasing and Publishing Packages" on page 240.
Edit URL	Allows you to edit the vendor's name and URL for publishing custom packages.
Delete URL	Deletes the select URL from the list of URLs.
Restore Defaults	Restores all settings to their default values.

To add a URL:

1. Click the **Add URL** icon .
2. Specify the vendor of the package and the URL where packages are available for download.
3. Click **OK**.

Package Based Policies

When a user modifies a policy object that belongs to a package, Designer marks the object as being customized. You can configure Designer to warn users that this occurs when they modify a package-based policy object. This setting is enabled by default.

To configure how Designer displays a warning when a user opens a policy that belongs to a package:

1. In the Preferences window in Designer, expand **NetIQ** > **Package Manager** and click **Package Based Policies**.
2. If you want to disable the warning, select **Do not prompt for policy customization on opening package based policies**.
3. If you want to enable the warning, select **Prompt for policy customization on opening package based policies**.
4. Click **OK**.

Vendor Information

Allows you to specify your vendor information for your packages in one location, instead of specifying the information each time you create a package. For more information, see "Creating Feature Packages" on page 229.

Table 19-49 *Preferences: NetIQ > Package Manager > Vendor Defaults*

Setting	Description
Vendor > Name	Specify the vendor name. If this is for internal consumption, specify the name of your company.
Vendor > Address	Specify the address for the vendor or your company.
Vendor > URL	Specify the URL of the vendor your company.
Vendor > eMail	Specify an e-mail for the vendor or your company.
Contact > Name	If there is a specific contact person for this package, specify the name.
Contact > eMail	If there is a specific e-mail address for the contact person, specify it in this field.

Provisioning

You can customize some Provisioning view behaviors by setting preferences. You access the preferences page through **Windows > Preferences > NetIQ > Provisioning**. The following table explains the settings on Provisioning preferences main page.

Table 19-50 *General Preferences*

Setting	Description
Prompt for deletion of User Application Configuration	When this option is selected and you delete a User Application from the Modeler, Designer asks whether to delete the provisioning objects on disk as part of the delete operation. By default, the provisioning objects are left on disk, even if the User Application is deleted.
Set delete from Identity Vault as default for all "Confirm Delete" dialogs	When you delete an object in the Provisioning view or the directory abstraction layer editor, you are prompted to confirm the deletion. This preference determines whether the check box labeled **Delete object in Identity Vault on deploy** in the confirmation dialog box is selected by default. Selecting this preference means the default is to delete the Identity Vault object. The local object is always deleted.
Show Provisioning View when new User Application is created or imported	Select this option if you want Designer to launch the Provisioning view when you create a new User Application driver or import an existing User Application driver.
Show Tooltips in Provisioning view	Select this option to enable (the default) tooltips in the Provisioning view.
Show Categories in Provisioning view	Select this option so Designer displays provisioning request definitions organized by category. You specify the category in the Overview panel. Categories are defined in the Provisioning Category list defined in the directory abstraction layer.
Show all localized e-mail templates	Select this option so Designer displays all localized e-mail templates as selectable options in the **E-Mail notification** tab. The Java language code is appended to the name of the e-mail template. For example, cn=Provisioning Notification Activity_es, cn=Default Notification Collection,cn=security indicates this is the Spanish language version of this template. When you select a localized template, that language is used regardless of the user's default language. When you select the default template (the template without a locale code), the e-mail is in the user's default language (if the default is a supported language).
Validate display names for supported locales	Select this option if Designer should validate display names. It ensures uniqueness of the display name within a locale, and that a display name is supplied (not blank) for each locale. Applies to display names defined by using the directory abstraction layer editor, provisioning request editor, or provisioning teams editor.
Prompt before performing query on Role Entitlement	When this option is selected, and you click **Run query in the Identity Vault**, Designer informs you that the query can take a long time to execute. It prompts to run the query or not. If this option is not selected, Designer runs the query and does not prompt you.

Setting	Description
Identity Vault Connection Timeout (in milliseconds)	The amount of time (in milliseconds) for Designer to connect to the Identity Vault. When it is set too low, you might encounter an error when setting Trustee Rights on a provisioning request definition or when trying to access the Identity Vault through the ECMA expression builder.

The following sections explain the additional preferences settings for provisioning:

- "Import/Deploy Preferences" on page 516
- "Migration Preferences" on page 517
- "Integration Manager" on page 517
- "Validation Mask Preferences" on page 517
- "Workflows Preferences" on page 517

Import/Deploy Preferences

Table 19-51 *Import/Deploy Preferences*

Setting	Description
Import > **Delete local object on import when object has been deleted in Identity Vault**	Select this option for Designer to delete local objects if the corresponding Identity Vault objects were deleted. This ensures that the Identity Vault and local files are in sync. Deselect this option if you want to leave the local files alone.
Import > **Prompt whether to overwrite runtime configuration on import from file**	Select this option if you are importing the driver from a test environment and want to deploy to a production environment. The User Application driver runtime relies on objects stored in the driver that you are not able to access in Designer. If you deploy a driver that does not contain these objects, it does not work properly. Deselect this option if you are importing the driver, modifying it, and deploying it back to the same driver set because the driver already has the runtime configuration objects.
Import > **Prompt to import role catalog**	Select this option if you want a prompt to appear to import role catalog.
Deploy > **Allow deployment of objects with validation errors**	Select this option if you want to deploy objects that fail validation checks. At deployment, Designer validates the definitions being deployed following the validation rules outlined in "Deployment and Versions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Administrator's Guide to Designing the Identity Applications*. Deselect this option to prevent deployment of definitions that fail validation. **WARNING:** Deploying objects that fail validation can result in errors in the User Application runtime.

Migration Preferences

Table 19-52 *Migration Preferences*

Setting	Description
Show warning about Identity Vault schema changes	When you select **Migrate**, Designer displays a dialog box warning you that schema changes (needed to support new features) must be made before you can deploy the migrated driver. If the updates have not been made, cancel the migration until they are complete. If you don't want to see this warning when you select **Migrate**, deselect this option.
Always deploy (un-deployed) User Application Driver	Applies to User Application drivers that have not been deployed to the Identity Vault (for example, User Application drivers imported from a driver configuration file). When you migrate an undeployed User Application driver, Designer prompts you to deploy the driver. Select the **Always deploy (un-deployed) User Application driver** option if you always want Designer to deploy the User Application driver, and do not want the dialog box displayed.
Show warning that editors will be closed	When you select the **Migrate** command, Designer warns you that all editors will be closed. Select this option if you don’t want this warning displayed each time you choose the **Migrate** command.

Integration Manager

The Integration Manager is used by the User Application workflow engine to provide Integration Activity support.

Validation Mask Preferences

Table 19-53 *Validation Mask Preferences*

Setting	Description
Validation Mask Table	Use this to define the validation masks available to form controls. Validation masks are regular expressions and must follow regular expression syntax. Designer provides a default set of validation masks. If they do not display validation masks in the form controls property sheets, enable them by clicking **Restore Defaults**, then clicking **Apply**.

Workflows Preferences

Table 19-54 *Workflow Preferences*

Setting	Description
Form Templates	Use this dialog box to remove or preview existing form templates.

Setting	Description
Diagram Preferences	Show Activity Id: Select this preference when you want the Workflow tab of the provisioning request definition editor to display the Activity IDs for each activity in the flow. Activity IDs are used by the ECMA expression builder and are written to the User Application's error logs. Show Flow Path Types: Select this preference when you want the Workflow tab of the provisioning request definition editor to display the Flow Path Types for each activity in the flow. Flow Path Types are used by the ECMA expression builder and are written to the User Application's error logs.

Validation

The Validation setting is an Eclipse setting that allows you to validate your project. For more details, see the Eclipse documentation (http://help.eclipse.org/kepler/index.jsp).

Table 19-55 *Preferences: Validation*

Settings	Descriptions
Allow projects to override these preference settings	Allows your project to override these preferences settings.
Suspend all validators	Allows you to suspend all validation actions that are performed on your project.
Save all modified resources automatically prior to validating	Saves any modified resource prior to running a validation. This option is not selected by default.
Show a confirmation dialog when performing manual validations	Allows you to display a confirmation dialog when performing a manual validation.
Selecting validators	The following validators run when a validation is performed. By default all validators are selected. ◆ DTD Validator ◆ HTML Syntax Validator ◆ MoudleCoreValidator ◆ XML Validator
Restore Defaults	Restores all of the settings back to the default values.

Web

The Web preference lets you specify how Designer should handle the editing and creations of CSS and HTML files.

- "CSS Files" on page 519
- "HTML Files" on page 520

CSS Files

The CSS Files preferences allow you to specify how Eclipse displays and manages CSS files. This is an Eclipse option; for more details, see the Eclipse documentation (http://help.eclipse.org/kepler/index.jsp).

Table 19-56 *Preferences: Web > CSS Files > Editor*

Setting	Description
Formatting: Line width	Specifies the number of characters in a line.
Formatting: Insert line break between properties	Specifies whether the editor should insert a line between the CSS properties.
Formatting: Disable wrapping in style attribute of HTML	Specifies whether the HTML editor (used in the e-mail notification template editor) should allow wrapping of the value of a style attribute.
Formatting: Indent using tabs > or spaces	Specifies how the first line of text indents.
Formatting: Indentation size	Specifies the size of the indent.
Formatting: Capitalization style	Specifies the default case for identifiers, property names, and property values.

- "Syntax Coloring" on page 519
- "Template" on page 519

Syntax Coloring

Table 19-57 *Preferences: Web > CSS Files > Editor > Syntax Coloring Settings*

Setting	Description
Syntax Element	Choose the content type for which you want to define a style.
Foreground/Background/Bold/Italic/ Strikethrough/Underline	Specifies the syntax highlighting and formatting for individual CSS elements.
Sample Text	Displays sample CSS with the selected syntax coloring options.

Template

Eclipse allows you to use a template file for the initial content of your cascading style sheets (CSS). The CS files are used to format the content in the Eclipse program. You can either create a new CSS file or import and existing CSS file through this page to use as a template. For more information, see the Eclipse documentation (http://help.eclipse.org/helios/index.jsp).

HTML Files

The HTML Files preferences allow you to specify how Designer displays and manages HTML files and content. This is an Eclipse option; for more details, see the Eclipse documentation (http://help.eclipse.org/helios/index.jsp).

Table 19-58 *Preferences: Web > HTML Files*

Preference	Description
Creating or saving files: **Line Delimiter**	Choices are: ◆ **Windows** ◆ **Unix** ◆ **Mac** ◆ **No translation**
Creating files: Add this suffix	Specifies the file suffix the editor should add when creating a new file. The default is `html`.
Creating files: Encoding	Specifies the editor's encoding for new files.
Loading files	Choose the encoding for files opened in the editor. Click **Use workbench encoding** to accept the default UTF-8, or select one from the list.

- "Editor" on page 521
- "Validation" on page 522

Editor

Table 19-59 *Preferences: Web > HTML Files > Editor*

Setting	Description
Formatting: Line Width	Specifies the number of characters for each line.
Formatting: Split multiple attributes each on a new line	Specifies what the editor should do with multiple attributes.
Formatting: Align final bracket in multi-line element tags	Specifies what the editor should do with final brackets
Formatting: Clear all blank lines	Specifies what the editor should do with blank lines
Formatting: Indent using tabs or spaces **Indentation size**	Specifies whether the indent should be using tabs or spaces, and also specifies the indentation size.
Content assist: Automatically make suggestions	Specifies whether to do automatic code completion.
Content assist: Prompt when these characters are inserted	Specifies the characters that initiate the content assist.
Preferred markup: Tag Names/Attribute Names	Specifies if the editor's suggestions should be in uppercase or lowercase.

- "HTML Styles" on page 521
- "HTML Templates" on page 522
- "Typing" on page 522

HTML Styles

Table 19-60 *Preferences: Web > HTML Files > Syntax Coloring*

Setting	Description
Syntax Element	Choose the content type for which you want to define a style.
Foreground/Background/Bold/Italic/ Strikethrough/Underline	Specifies the syntax highlighting and formatting for individual CSS elements.
Sample Text	Displays sample CSS with the selected syntax coloring options.

HTML Templates

Table 19-61 *Preferences: Web > HTML Files > Templates Settings*

Setting	Description
Templates	The templates are used in the code completion in the source editor. Use this preference to add, remove or edit templates.

Typing

Table 19-62 *Preferences: Web > HTML Files > Typing*

Settings	Descriptions
Automatically close: Comments	The HTML editor automatically closes any comments added to the HTML file.
Automatically close: End tags	The HTML editor automatically closes any end tags in the HTML file.
Automatically remove: End tags	The HTML editor automatically removes any end tags when creating empty self-closing tags.

Validation

Allows you to define how the HTML editor validates the HTML markup. You can set each validation to a warning, error, or to ignore the problem. You can set these options for the following items:

- Elements
- Attributes
- Document Type
- Comments
- CDATA Sections
- Processing Instructions
- Entity References
- Text Regions

XML

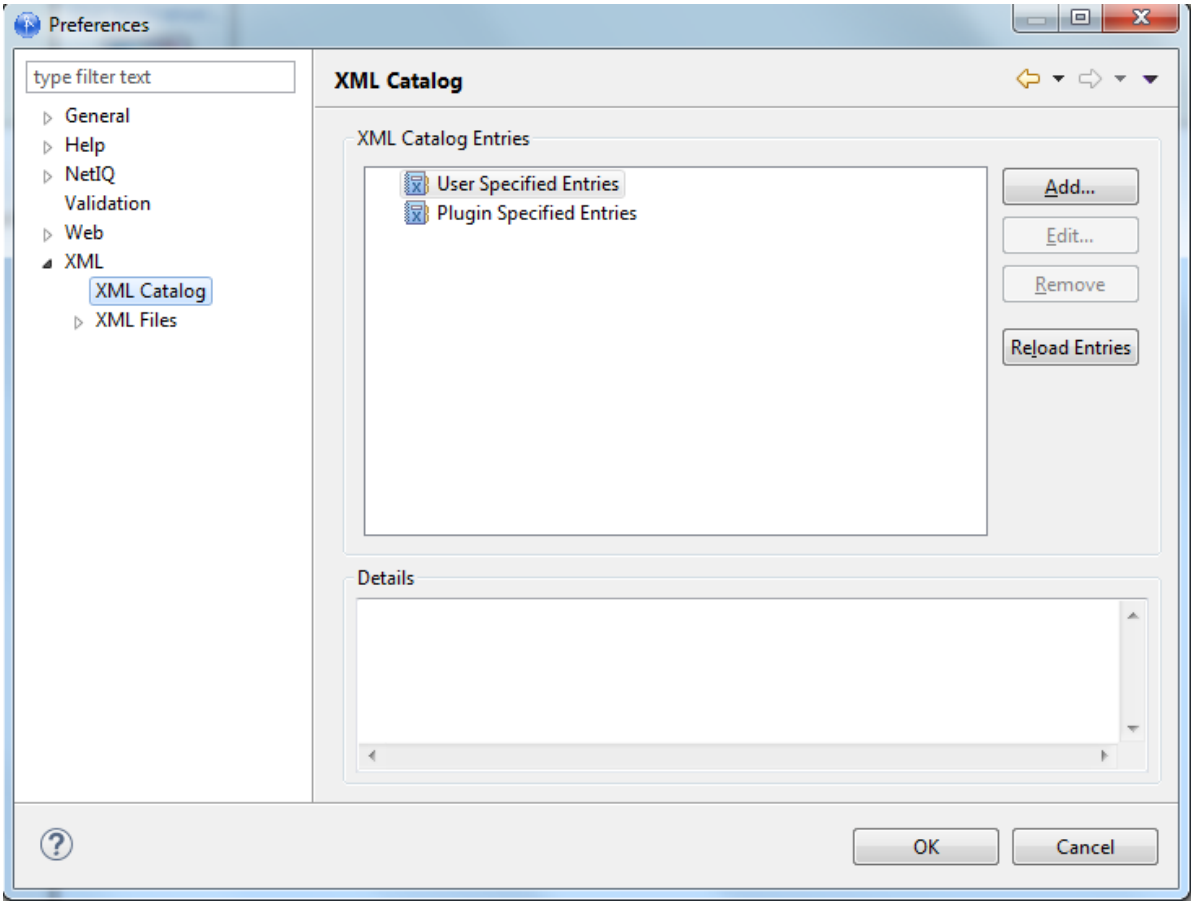
The XML preferences lets you specify how Designer should handle editing and creation of an XML catalog and XML files. This an Eclipse option; for more details, see the Eclipse documentation (http://help.eclipse.org/kepler/index.jsp).

- "XML Catalog" on page 523
- "XML Files" on page 524

XML Catalog

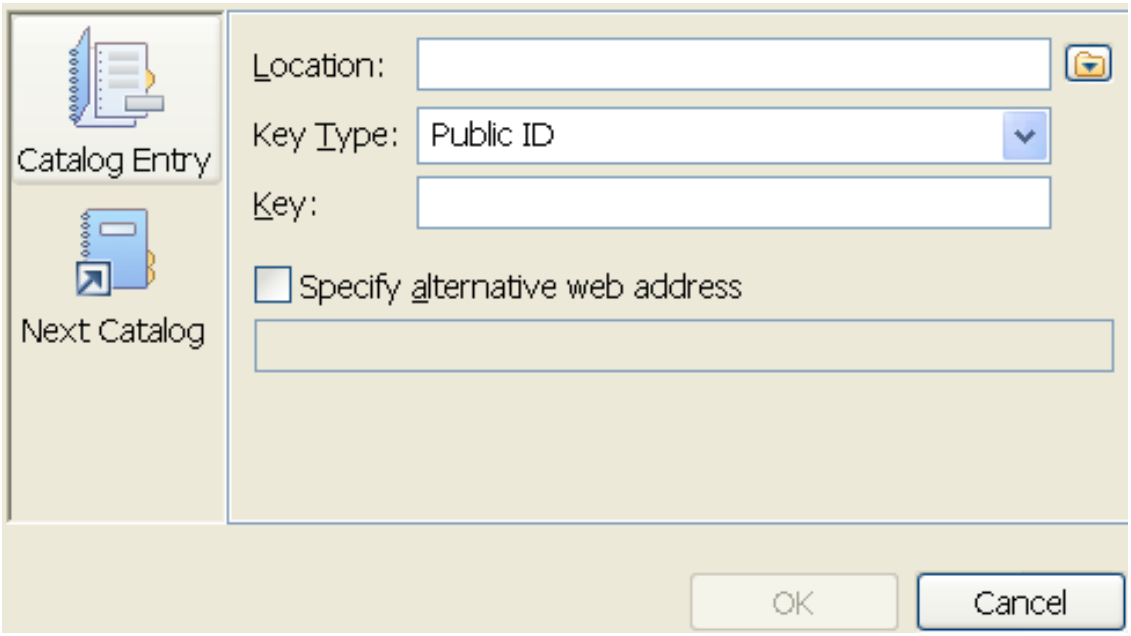
The XML Catalog preferences allow you to manage the WST XML catalog implementation. You can add, edit, or delete user-specified catalogs. You cannot use this preference to manage the plug-in specified entries.The XML editor uses the WST XML catalog implementation to resolve XML schema and DTD references for associating URLs, system, and public identifiers with URLs.

Figure 19-5



To add a user-specified entry:

1 Click Add.



2 Fill in the fields as follows:

Field	Description
Location	Specify a location on disk or a URL of the schema or DTD. Use the Search icon to search Designer's workspace or the file system.
Key Type	Specify the key type. Values are public identifiers for DTDs or URIs for XML schemas.
Key	Specify a unique key.
Specify alternative web address	Optionally, specify an alternative Web address for locating the schema or DTD.

3 Click OK to save.

XML Files

You can set the following general XML File preferences:

Table 19-63 *Preferences: XML > XML Files*

Setting	Descriptions
Creating files: Add this suffix	Add a suffix to the file. The default is XML.
Creating files: Encoding:	Select the encoding used by the user.
Creating files: IANA	The IANA name is used in the encoding statement of the XML file.
Validating files: Indicate when no grammar is specified	Specifies whether to display a warning when no grammar (such as XML Schema or DTD) is associated with the XML document.
Validating files: Process XML Inclusions	If the XML file contains inclusions (snippets from an HTML file used to create the dynamic HTML page), process these inclusions.

Editor

Table 19-64 *Preferences: XML > XML Files > Editor*

Category	Preference	Description
Formatting	Line width	Specifies the number of characters in a line. The default is 72.
	Split multiple attributes each on a new line	Specifies how attributes are formatted (whether to show each attribute on a separate line).
	Align final bracket in multi-line element tags	Allows you to align the final bracket ">" in multi-line element tags.
	Preserve whitespace in tags with PCDATA content	Specifies whether to preserve any white spaces that are in tags containing PCDATA content.
	Clear all blank lines	Specifies whether blank lines are removed when formatting.
	Indent using tabs/ or spaces	Specifies whether to use tabs or spaces as indentation and indentation size.
	Indentation size	
Content Assist	Automatically make suggestions	Specifies whether to do automatic code completion.
	Prompt when these characters are inserted	The list of characters that initiate code completion.
	Suggestion strategy	Specifies whether to use Lax or Strict grammar when making suggestions
Grammar constraints	Use inferred grammar in absence of DTD/Schema	Specifies whether to display code completion suggestions based on existing content of the XML document.

- "Syntax Coloring" on page 526
- "XML Templates" on page 527
- "Typing" on page 527

Syntax Coloring

The XML syntax coloring lets you specify the syntax highlighting (foreground and background color) and the text formatting for individual XML constructs.

Table 19-65 *Preferences: XML > XML Files > Syntax Coloring Settings*

Setting	Description
Syntax Element	Choose the content type for which you want to define a style.
Foreground/Background/Bold/Italic/ Strikethrough/Underline	Specifies the syntax highlighting and formatting for individual CSS elements.
Sample Text	Displays sample CSS with the selected syntax coloring options.

XML Templates

Use the XML Templates preference page to define XML templates. The templates are used in the code completion in the XML Source editor. For example, selecting the XSL `Processing Instruction` template in the code completion inserts `<?xml-stylesheet type="text/xsl" href="?">` in the source editor and places the cursor in the `href` value.

Typing

Table 19-66 *Preferences: XML > XML Files > Typing*

Settings	Descriptions
Automatically close: Comments	The HTML editor automatically closes any comments add to the HTML file.
Automatically close: End tags	The HTML editor automatically closes any end tags in the HTML file.
Automatically remove: End tags	The HTML editor automatically removes any end tags when creating empty self-closing tags.

20 Best Practices

This section includes some tips and best practices for using Designer.

- To enhance Designer performance, remove any unused packages from the package catalog. To do so, right-click package catalog and then select **Remove Unused Packages**. For more information, see "Removing Packages from the Package Catalog" on page 191.
- Retain only the required packages in Designer. To do so, use the **Manage Packages** option in Designer. For more information, see "Managing Installed Packages" on page 181.
- Have minimal projects in the workspace. Ensure to close or disable the projects with User Application Driver on which you are not working on.
- When you are working on User Application objects, close the Outline view and use the Provisioning view. When you are not working on User Application objects, close the Provisioning view and use the Outline view.
- Open only limited number of editors in Designer, to minimize the memory consumption. Alternatively, you can increase the default maximum memory settings in the `Designer.ini` file.
- In the Outline View, collapse all drivers and objects that you are not working on. This improves the Designer performance while opening a new editor or a policy.
- Run the Garbage Collector when Designer is slow and the memory consumption is high. For more information, see "Freeing Heap Memory" on page 540.
- Before you import data into an empty project, it is recommended that you navigate to **Windows > Preferences > NetIQ > Identity Manager > Import/Deploy > Behaviors > Summary Dialog** and de-select the **Show the summary dialog prior to performing an import** check box. This improves Designer performance while importing data. After the data import process is completed, you must select the **Show the summary dialog prior to performing an import** check box again.
- Do not use the special character * in the server context in iManager. When the server context has * in it's name, and the project containing that server is imported into Designer, Designer does not import the project as intended. Instead, Designer displays two entries with the same name.
- If you want to delete a form, ensure that you close the form before proceeding with the delete operation.
- Before replacing a string in Designer using the **Search/Replace** operation, ensure that you resolve any discrepancies manually in the XML file. The Replace operation does not validate the XML file before performing the operation.

21 Troubleshooting Designer

- "Running the Project Checker" on page 532
- "Viewing the Error Log" on page 532
- "Turning on Trace Messages" on page 534
- "Checking Loaded Plug-Ins" on page 535
- "Deploying OpenText Identity Manager Objects" on page 536
- "Display Issues" on page 538
- "Freeing Heap Memory" on page 540
- "Project Files Are Not Encrypted" on page 541
- "Users Cannot Import and Check In Multiple Instances of the Same Package Under Version Control" on page 541
- "Drivers Not Associated with Base Packages After Live Import" on page 541
- "Error Messages and Solutions" on page 543
- "Reporting Bugs and Giving Feedback" on page 554
- "Designer Does Not Deploy the Global Configuration Attribute on a Driver or a Driver Set" on page 555
- "Designer Takes Too Long To Deploy a Driver or a Driver Set" on page 555
- "Manually Removing Invalid Designer Shortcut from mac Launchpad" on page 555
- "Unable to Launch Designer Application on Mac" on page 556
- "Designer Fails to Deploy Driversets of Mixed Versions of OpenText Identity Manager" on page 556
- "Designer Does Not Respond If The Project Name Contains a Space" on page 556
- "Designer Fails on Linux to Open Data Item Mapping of a Workflow Activity Containing Non-ASCII Characters" on page 557
- "Importing a Workflow Created Using Older Version of User Application Driver" on page 557
- "Form Builder Does Not Launch Properly From Designer on Linux Platforms" on page 557
- "Unable to Deploy Large PRDs" on page 558
- "Unable to Connect to the Git Repository After Committing Package Changes for the Second Time" on page 558
- "Unable to Launch Form Builder on Linux Platforms" on page 559
- "Modifying The src Attribute Used in a Policy" on page 559
- "Modifying the do-create-resource Action Manually to Use REST API" on page 560

Running the Project Checker

Designer provides a Project Checker tool to check your project. The project can be checked at any time, but you should run the Project Checker before deploying your project. The Project Checker checks for proper design, contexts, server associations, policies, missing user data, and dependency problems that would cause the deployment of project into the Identity Vault to fail. It only checks the objects in Designer; it does not check the current objects in the Identity Vault.

To learn more about the Project Checker, see "Checking Your Projects" on page 429.

Viewing the Error Log

If something isn't working, messages written to the error log might help you. The log is named `.log`. It is a hidden file.

To view the error log, you can use menus or browse the file system.

- "Browsing the File System" on page 532
- "Using Menus" on page 532
- "Event Details" on page 533
- "Customizing Filter Settings" on page 533

Browsing the File System

1. Browse to your Designer workspace.

   In Windows, the log file is typically in subfolders in the `/eclipse/workspace/.metadata` directory.

   In Linux, the log file is typically in the Home directory, in the `/eclipse/workspace/.metadata` directory.
2. Open the log file.

Using Menus

1. Select **Window** > **Show View** > **Other** > **PDE Runtime** > **Error Log**.
2. Click **OK**.

If you view the log through the application, a list of messages displays. For a description of the icons located in the upper right corner of the Error Log view, see "Error Log View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

The following options are available when you right-click inside the Error Log view:

Table 21-1 *Right-Click Options in the Error Log View*

Operation	Description
Copy	Enables you to copy event details to the clipboard.
Clear Log Viewer	Clears all the entries in the Error Log viewer.
Delete Log	Deletes all items in the Error Log.
Open Log	Opens an error log entry.
Restore Log	Enables you to restore log entries that have been previously cleared.
Export Log	Enables you to export the Error Log to a location on the file system.
Import Log	Enables you to import a file from the file system to the Error Log.
Event Details	Opens the Event Details window.

To sort messages in the Error Log view, click the appropriate header bar.

Event Details

To view event details, double-click an error log message or right-click an error log message, then click **Event Details**.

The following options are available in the Event Details window:

Table 21-2 *Event Details Window*

Operation	Description
Date	Displays the date and time the error occurred.
Severity	States the severity of the error.
Message	Displays the message of the error.
View Details of Previous Event	Up and down arrows that enable you to scroll through the event details of each event in the error log.
Copy	Enables you to copy event details to the clipboard.
Exception Stack Trace	Displays Exception Stack Trace (if available).
Session Data	Provides relevant session data.

Customizing Filter Settings

To access the Log Filters window:

1. On the Error Log view toolbar, click the **Menu** icon.
2. Click **Filters**.

The following options are available in the Log Filters window:

Table 21-3 Log Filters Window

Operation	Description
Event Types	Set what type of information you want displayed in the error log. The error log can be configured to display any combination of Information, Warnings, and Errors.
Limit Visible Events	Set a limit on how many events you want displayed in the error log at one time.
Show Events Logged During:	Specify whether to show events logged during all sessions, or your most recent session.

Turning on Trace Messages

You might want to send trace messages to the error log so that the messages are captured in a file. You can then easily e-mail the trace message to OpenText Support or others.

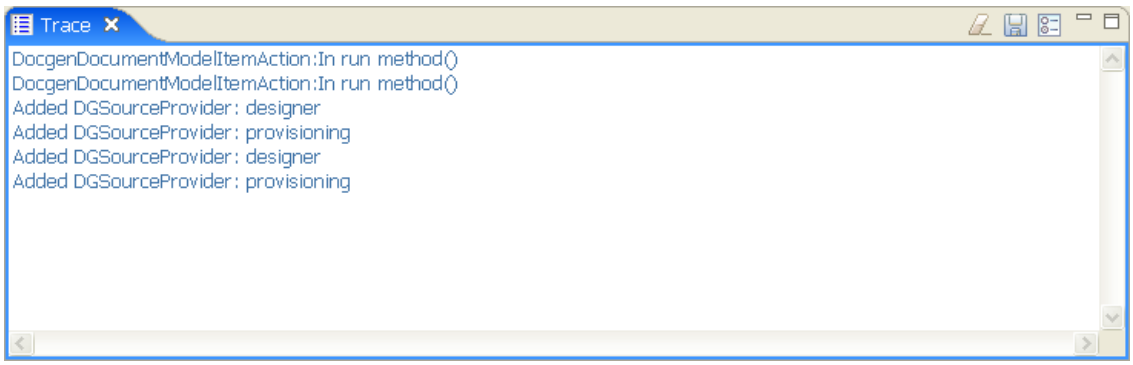
Programmers sometimes place hidden messages in their code so that if you are having problems, you can turn on the trace functionality and get additional insight. Even if you don't understand the hidden messages, they can help OpenText Support diagnose the problem.

To get trace messages:

1. Click **Window** > **Preferences** to display the Preferences dialog box.
2. Click **NetIQ** > **Designer** > **Trace**.
3. Select **Enable tracing**, then select the options that you want to include or show.
4. Select the plug-ins that you want to trace, then click **OK**.

To view the results of traces:

1. Select **Window** > **Show View** > **Trace**.
2. View data in the Trace view.



3. You can also turn on trace options from the Trace view by clicking the Preferences icon in the Trace view.

The following options are available when you right-click inside the Trace view:

Table 21-4 *Right-Click Options in the Trace View*

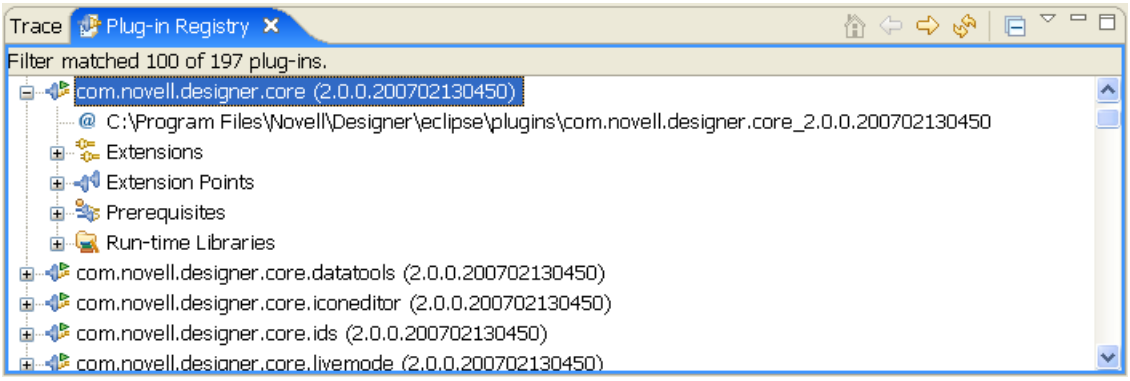
Option	Function
Undo	Undo a previously executed action.
Cut, Copy and Paste	Cut, copy and paste items in the Trace view by selecting the item, then clicking the desired action.
Delete	Delete items in the Trace view.
Select All	Simultaneously select all trace messages in the Trace view.

An icon toolbar is located in the upper right corner of the Trace view. For information about the icons in this toolbar, see "Trace View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.

Checking Loaded Plug-Ins

A problem can occur if a plug-in fails to load. To see which plug-ins are loaded:

1. Select **Window** > **Show View** > **Other**.
2. Open the PDE Runtime folder.
3. Click **Plug-in Registry** > **OK**.



   The Plug-in Registry page lists the Designer plug-ins, which have a green triangle in the plug-in icon.
4. Use the **Home** icon to bring you to the top of the plug-in list.
5. Select a plug-in, then use the right-arrow icon to drill into the plug-in and use the left arrow icon to return.
6. Use the **Refresh** icon to refresh the Plug-In Registry view.
7. Use the Plug-In Registry view toolbar to select **Show Active Plug-Ins Only**.

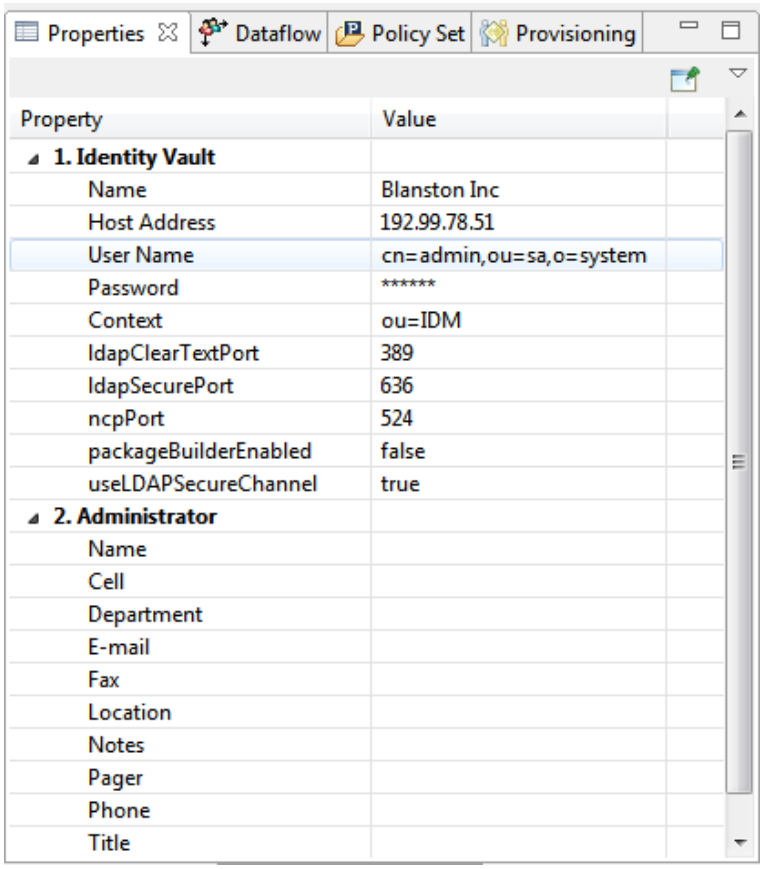
Deploying OpenText Identity Manager Objects

When you see an error message in Designer, the message corresponds to the place where Designer could not complete the task, and indicates the best place to start troubleshooting. This section discusses the common problems you face when deploying OpenText Identity Manager objects into an eDirectory tree. To see error messages and possible solutions, see “Error Messages and Solutions” on page 543.

Deployment Considerations

- Ensure that the OpenText Identity Manager server meets the system requirements necessary to run OpenText Identity Manager. See the Overview chapter in the OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Linux or OpenText™ Identity Manager CE 24.4 (v4.10) Install and Upgrade Guide for Windows for requirements.
- Ensure that the OpenText Identity Manager server you are deploying to has OpenText Identity Manager installed and holds a real copy of the objects to which you want to synchronize. The server running OpenText eDirectory must have a Master Read-Write or a Filtered Read-Write replica.
- Ensure that the Java software installed on the server is running correctly, because OpenText Identity Manager is dependent on Java. If Java is corrupted, you might be able to deploy to a OpenText Identity Manager server but not run the OpenText Identity Manager drivers.
- To deploy an OpenText Identity Manager-based project or an object in a project, you must have access to the OpenText eDirectory tree that is associated with the Identity Vault you are designing. Select the Identity Vault you want to deploy, then look in the Properties view below the Project/Outline view.

Figure 21-1 *The Properties View*



In the Properties view, ensure that the Identity Vault's Name, Host Address, User DN, Password, Deploy Context's Distinguished Name (DN), and Identity Vault information is complete and accurate. (You can click the **Browse** icon to find the Deploy Context's DN on an existing tree if the other information is accurate and Designer can attach to the OpenText eDirectory tree.) You need this information to deploy anything, even a policy, into an existing OpenText eDirectory tree running the OpenText Identity Manager engine.

- Use the Deploy feature only after you have thoroughly tested the rules and policies that make up your drivers. To test a policy, use the Policy Simulator (right-click a policy and select **Simulate**, then click **Start** to see the simulation results of the policy that is being tested). For policy design, see the Policy Builder Help topics within the Designer utility.

  You can use the Import feature to import a driver, a channel, or a policy. You can then modify the object or objects, run the Policy Simulator to ensure that the object is working correctly, then deploy the object back into the test tree for further analysis. You can also run the Compare feature to see the differences between your modified driver and the driver that is currently running on an Identity Vault server.

- In the Outline view in the Project Group view, right-click the driver object in question (you can also double-click the driver object). Use the Properties window to make most changes to drivers. Properties are unique to each driver.

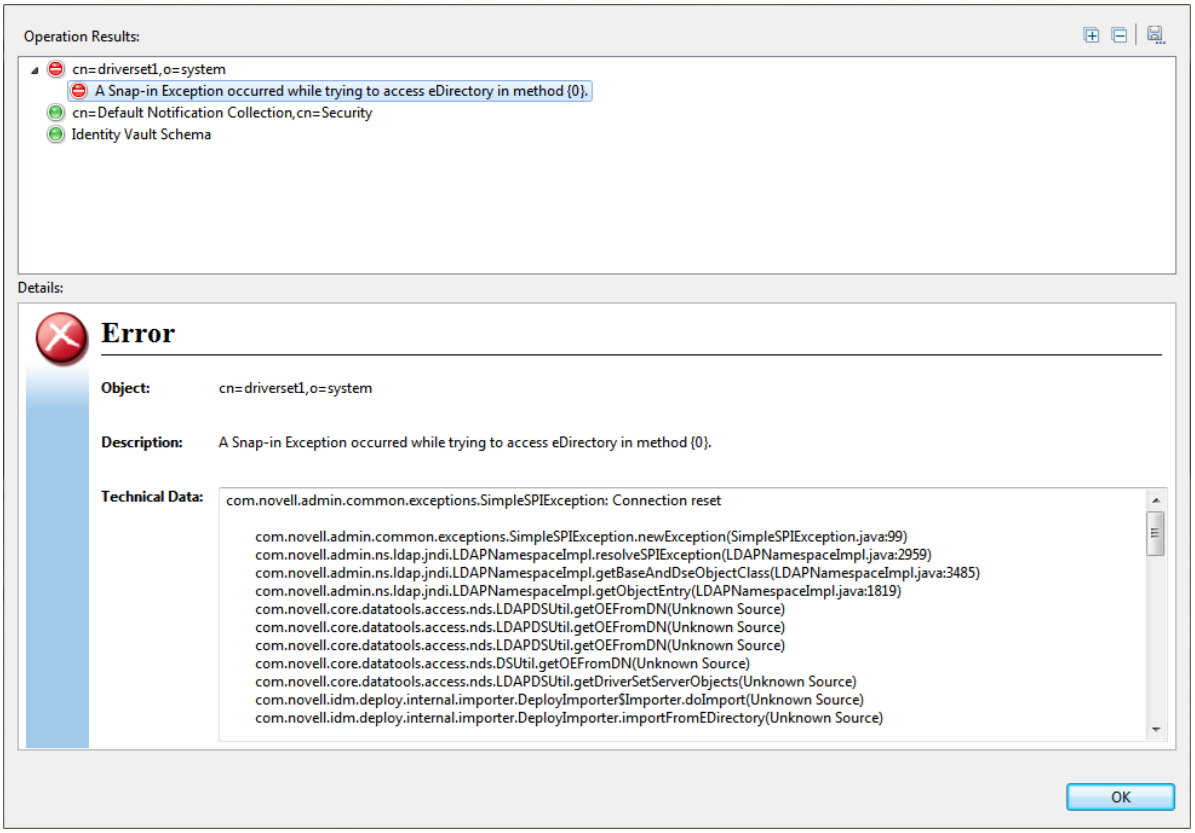
  A simple driver problem is specifying the incorrect context (Distinguished DN) for an OpenText eDirectory tree. For example, the context of a user object in OpenText eDirectory is shown with the slash notation (for example, **Blanston\Sales\Users**) on the Properties of the OpenText Identity Manager driver or when you import the driver. However, different drivers can use

formats other than the slash notation. For example, Active Directory and LDAP drivers use comma-delimited format (OU=Users,OU=Sales,O=Blanston). See the driver guides for further details on the drivers you are deploying.

An Example Deployment Error

When you deploy an Identity Vault for the first time, there are several common sources for errors, from incorrectly typing information to not completing the driver set templates.

Figure 21-2 *Default Server Container Message*



Right-click the Identity Vault in the Modeler view, select **Properties > Server List**, then click the **Edit** icon to edit the server information.

Display Issues

The following sections include display issues users may encounter when using Designer.

No F1 Help in Maximized Editors

Context-sensitive help is available when you press F1. However, if you maximize an editor (for example, the Modeler), help topics do not display when you press F1. To view the help, minimize the editor.

Using 120 DPI Fonts in Windows

120 DPI is too large for text in standard Windows XP decorations. Adjust the display settings:

1. In the Control Panel, select **Display > Appearance > Effects**.
2. In **Use the following method to smooth edges of screen fonts**, toggle **Standard** to **ClearType**.

   If you have a display that needs 120+ DPI fonts, you need ClearType. In addition to the obvious anti-aliasing aspects, ClearType provides better weight to the fonts. Without ClearType, the fonts are too thin and light, decreasing readability.
3. Click **OK**, then click **Advanced.**
4. In the **Item** field, reduce the **Icon, Menu, Message Box**, **Selected Items**, and **ToolTip** sizes.
5. Reduce the title bars and related controls to a preferred size.
6. Fix the icon spacing and scroll bar width.
7. Make sure that the display is set at a high resolution.

   This helps eliminate most of the display related issues on an HD monitor.

Display Issues on Linux

- "GNOME" on page 539
- "KDE" on page 539

GNOME

If you encounter display issues in GNOME:

1. Select the **Applications** menu.
2. Click **Preferences** > **Font**, then decrease the size of the application font.
3. You can also adjust the thematic elements to your liking.

   Keep in mind that GTK thematic elements can cause performance issues with Designer. If Designer is running slowly, especially when you use pull-down menus and other widgets, you might try changing to a simplified GTK theme.

   Normally, this process fixes display issues.

KDE

Because Eclipse (Designer) is a GTK application, you should use GTK themes instead of qt-based themes.

First, you need to prepare to use the themes.

You must remove the gtk-qt-engine package. This can be done through YaST or by using the instructions given in "Running Designer on Linux with gtk-qt-engine," in the Identity Manager 4.6 Designer Release Notes (https://www.netiq.com/documentation/identity-manager-46/releasenotes_designer46/data/releasenotes_designer46.html).

You need to have the following packages installed on your Linux system. If you installed the GNOME subsystem, you already have these packages installed:

- gtk-engines
- gtk2-engines
- control-center2 > Gnome Control Center
- gtk2-themes > or the themes you downloaded, and all the related dependencies
- gnome-themes is only needed if you are going to use Gnome Control Center to set your theme

After you have completed the prerequisites, do one of the following:

- Set your GTK theme and font settings from the KDE SUSE menu. **Select Utilities > Desktop > Gnome Control Center**. You can set this control center application to automatically run each time KDE is started. The following command accomplishes this:

  ```
  ln -s /opt/gnome/lib/control-center-2.0/gnome-settings-daemon /home/user/.kde/Autostart
  ```

  For **user**, use your username.
- Create a GTK control file (usually named `.gtkrc-2.0`) in your user home directory or the directory where your system is configured to look for GTK2_RC_FILES. Entering `set |grep gtk` shows how this environment variable is configured and the files it requires. You can use any font and GTK theme that you prefer.

  For example:
  ```
  include "/opt/gnome/share/themes/Xfce-stellar/gtk-2.0/gtkrc"
  style "user-font"
  /gtkrc" style "user-font"
  {
  font_name="Sans Serif 6"
  }
  widget_class "*" style "user-font" gtk-theme-name="Xfce-stellar"
  gtk-font-name="Sans Serif 6"
  ```

Copying, Pasting, and Dragging in the Navigator View Don't Update Version Control

Copying and pasting or dragging and dropping operations in the Navigator View are not handling files properly if the files are under version control. The workaround is to perform these operations from the Project view.

Freeing Heap Memory

A status field at the bottom of Designer displays heap memory used and heap memory available for an application or other item in Designer.

The information varies, depending on which item (for an example, an application) you click in the Modeler, Outline view, or other editors.

To free unused heap memory at any time, click the **Run Garbage Collector** icon.

Project Files Are Not Encrypted

Passwords are obfuscated. However, if you have other sensitive data in your project file, it is not encrypted in any way, and you must take care to safeguard your information.

Users Cannot Import and Check In Multiple Instances of the Same Package Under Version Control

In an environment where you use Designer with Subversion for version control, if a user creates a new driver, imports the required driver packages into the Designer project, and checks the driver and driver packages into Subversion, and then another user tries to create a new driver that imports a package already installed with the first driver, Designer returns the following error:

```
Unable to check in package 'PackageName' (Version). A package with that
version is already under version control.
```

A single user should install and check in a particular package or set of packages. After the first user installs and checks in a package, other users can then use that package and check in their changes.

If you encounter the error message above, you must revert the Package Catalog in Subversion and then manually re-import the new packages to resolve the issue.

For more information about best practices for managing packages with Subversion, see "Managing Packages Best Practices" on page 471.

Drivers Not Associated with Base Packages After Live Import

If you upgrade to Designer and perform a live import of a package-based Identity Vault configuration, the Properties page of one or more drivers may not display the base package for that particular driver. This indicates that the driver is not associated with its base package.

To configure Designer correctly, you must manually associate the appropriate base package with the driver:

1. In Designer, navigate to the Modeler view.
2. Right-click the imported driver and select **Driver** > **Properties**.
3. In the Properties window, click **Packages**.
4. Click the plus icon.
5. In the Select Packages window, select the appropriate base package for the driver.

NOTE: To determine the appropriate base package for a driver if the Select Packages window displays multiple versions of the same base package, you can refer to the pre-upgrade Designer workspace for the correct version number.

If your previous Designer workspace is unavailable, select the earliest version available for the version of OpenText Identity Manager with which the driver was installed. You should then upgrade to the latest version of the base package.

6 Select **Associate base package without complete install** and click **OK**.

7 Repeat Step 2 through Step 6 for each imported driver.

For information about the base packages installed with Designer 4.0 and 4.0.1, see Table 21-5.

Table 21-5 *Base Packages Installed in Designer*

Base Package Name	Base Package Short Name	Released Versions
Data Collection Service	NOVLIDMDCSB	1.0.0
		1.0.4
Driver for Active Directory	NOVLADBASE	1.0.0
		1.0.1
		1.0.3
Driver for Avaya PBX	NOVLAVYAB	1.0.0
Driver for Blackboard	OBNDBKBDBASE	1.0.0
Driver for Delimited Tex	NOVLDTXTBASE	1.0.0
Driver for eDirectory	NOVLEDIRBASE	1.0.0
Driver for Google Apps	NOVLGGLEBASE	1.0.0
		1.0.1
Driver for GroupWise	NOVLRSERVB	1.0.0
		1.0.1
		1.0.2
Driver for JMS	NOVLJMSBASE	1.0.0
Driver for LDAP	NOVLLDAPBASE	1.0.0
Driver for Lotus Notes	NOVLNOTEBASE	1.0.0
		1.0.1
Driver for PeopleSoft	NOVLPSFTB	1.0.0
Driver for RSA	TRVRRSABASE	1.0.1
Driver for SalesForce.com	NOVLSFBASE	1.0.0
Driver for SAP Business Logic	NOVLSAPBLB	1.0.0
		1.0.1
Driver for SAP HR	NOVLSAPHRIB	1.0.1
		1.0.2

Base Package Name	Base Package Short Name	Released Versions
Driver for SAP Portal	NOVLPORTB	1.0.0
		1.0.1
Driver for SAP User (JCo3)	NOVLSAPUBASE	1.0.0
		1.0.1
		1.0.2
Driver for Sentinel	NOVLSENTB	1.0.0
Driver for SharePoint	NOVLSPNTBASE	1.0.0
		1.0.1
		1.0.2
		1.0.3
Driver for SOAP	NOVLSOAPBASE	1.0.0
Driver for SunGard Banner	NOVLBNNRBASE	1.0.0
Driver for Work Order	NOVLWOBASE	1.0.0
Entitlements Service Driver	NOVLRBEBASE	1.0.0
		1.0.1
ID Provider Driver	NOVLIDPROVB	1.0.0
Loopback Driver	NOVLLBACKB	1.0.0
Managed System Gateway	NOVLIDMMSGWB	1.0.1
Null Service Driver	NOVLNULLBASE	1.0.0
Role Service Driver	NOVLRSERVB	1.0.0
		1.0.1
User Application 4.0 Driver	NOVLUABASE	1.0.1
		1.0.2
User Application 4.0.1 Driver	NOVLUABASE	1.0.5

Error Messages and Solutions

When you see an error message in Designer, the error message corresponds to the place where Designer could not complete the task and indicates the best place to start troubleshooting. This section discusses the error messages you might see when deploying OpenText Identity Manager objects into an eDirectory tree, followed by their cause and possible solutions.

- "Identity Vault Configuration Errors" on page 544
- "Driver Configuration Errors" on page 544
- "Internal Designer Errors" on page 545

- "eDirectory Access Errors" on page 546
- "OpenText eDirectory Object/Attribute Creation Errors" on page 547
- "Warnings" on page 549

Identity Vault Configuration Errors

```
Cannot connect to host [Identity Vault Host]; verify the address is correct
and that the server is running.
```

Possible Cause: The address listed in the Identity Vault properties is incorrect or the server is not running.

Solution: Verify that the server address is correct and that the server is up and running.

```
[User] could not be authenticated to [Identity Vault Host]. Cannot proceed.
```

Possible Cause: The username or password listed in the Identity Vault properties is incorrect.

Solution: Verify the username specified in the Identity Vault properties and reenter the user's password.

Driver Configuration Errors

```
The driver configuration file [Driver Config File] is not a valid XML
document: [Error Message].
```

Cause: The Driver Configuration file being imported from the file system does not contain a valid XML document.

Solution: Fix the Driver Configuration file format.

```
The XML contained the file named [Driver Config File] is not a driver
configuration file. The file cannot be imported.
```

Cause: The Driver Configuration file being imported from the file system is a valid XML document but is not a valid driver configuration file.

Solution: Import a driver configuration file.

```
The following 'XML DOM Exception' was thrown.
[ExceptionInfo]
```

Cause: The Driver Configuration XML document is incorrectly formatted. This is probably an internal error because driver configuration files are dynamically generated by Designer for deployment.

Solution: Turn on trace for Designer. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**. Attempt to deploy again, then send the trace file to OpenText Support.

```
The following 'Number Format Exception' was thrown.
[ExceptionInfo]
```

Cause: An integer value in the driver configuration file being deployed is invalid. All integer fields in Designer should validate the content when it is set.

Solution: Turn on trace for Designer. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**. Deploy again and analyze the generated driver configuration file to see if all integer attribute values are correct. Identify the incorrect parameter in Designer, correct the setting, and redeploy.

`The specified driver configuration file does not contain a valid driver configuration.`

Cause: Designer attempted to process a dynamically generated driver configuration file with an invalid format.

Solution: Turn on XML tracing for the Import/Deploy plug-in. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**. Deploy again, then send the trace to OpenText Support. Otherwise, edit and correct the configuration file being imported.

`Tree population is not supported from a Driver Set configuration. Tree population components will be ignored.`

Cause: The driver configuration file being processed has a `<ds-object>` element under a `<driver-set-configuration>` element, which is not permitted.

Solution: If this is a dynamically generated configuration file, contact OpenText Support; otherwise, move the `<ds-object>` element under a `<driver-configuration>` element.

`The following Driver Set based global variables could not be resolved: [Global Variable List]`
`These variables exist in both the source and target Driver Sets. The two definitions, however, have different types.`

Cause: The driver configuration file being processed has global variable definitions that could not be resolved.

Solution: If this is a dynamically generated configuration file, contact OpenText Support. If it is a driver configuration file on disk, check the global variable definitions.

`The driver configuration file being processed does not contain a valid driver configuration.`

Cause: The driver configuration file being processed does not contain a `<driver-configuration>` element.

Solution: If this is a dynamically created configuration file, turn on XML tracing for the Import/Deploy plug-in. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**. Deploy again, then send the trace to OpenText Support. Otherwise, edit and correct the configuration file that is being imported.

`The specified driver configuration file was only intended to be imported from a ConsoleOne command line.`

Cause: The driver configuration file being processed is not a valid document.

Internal Designer Errors

`An internal error has occurred in the Designer Data Model: The policy named [Policy Name] does not know its container.`

Possible Cause: The policy being deployed is not contained in a Channel or Driver object. This is an abnormal error, indicating that the Designer model has become corrupted.

Solution: Contact OpenText Support.

`Error Thrown while building Server Specific DsAccess for: "<serverName>"`

Possible Cause: The Network Address is not configured properly for that host address.

Solution: Ensure that the LDAP ports are configured only for the respective host IP. The command to set the LDAP interfaces to the respective Host IP: `ldapconfig set "ldapInterfaces=ldap://<hosted IP:port>,ldaps://<hosted IP:port>" -a <username> -w <password>`

eDirectory Access Errors

```
The following 'Component Creation Exception' occurred while trying to
access eDirectory.
[Exception Info]
```

Cause: A value contained in the driver configuration file being deployed could not be successfully created in eDirectory. This is probably an internal error because driver configuration files are dynamically generated by Designer for deployment. However, if the Driver in Designer was created by importing a driver configuration file from the file system and that configuration file contained a Tree Population Segment, a value within a `<ds-object>` element might be invalid.

Solution: Turn on trace for Designer. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**. Deploy again and analyze the generated driver configuration file to see if any `<ds-object>` elements exist. If they do, verify that all attribute values are correct. If no `<ds-object>` elements exist or if all values seem to be correct, contact OpenText Support.

```
The following 'IO Exception' occurred while trying to access eDirectory.
[ExecptionInfo]
```

Cause: This is a Java exception indicating that Designer could not perform the requested input or output operation.

Solution: Contact OpenText Support.

```
DSAccessException:
[ExceptionInfo]
```

Cause: Designer could not connect to the target deployment server.

Solution: Verify that the server information specified in the Identity Vault properties page is correct and that the eDirectory server is up and running.

```
The following 'Namespace Exception' occurred while trying to access
eDirectory. ({0})
```

Cause: This is a namespace exception indicating that there is a problem with the eDirectory schema, such as a missing attribute or class.

Solution: Verify that the eDirectory schema being imported from or deployed to is correct. If the driver being deployed contains Tree Population segments, verify that the objects being created are valid for the target eDirectory schema.

`An exception occurred during the deployment. Cannot perform the operation.`

Cause: An unknown exception was encountered.

Solution: Contact OpenText Support.

```
The following 'Snapin Exception' occurred while trying to access
eDirectory.
[ExceptionInfo]
```

Cause: Snap-in exceptions can be thrown in certain methods to report exceptions or errors during import/deploy. Subclasses of a snap-in exception include:

- NotAContainerException: There was a call to get the children of an OpenText eDirectory object that is not a container.
- ObjectNotFoundException: The object being resolved cannot be found in OpenText eDirectory.
- SPIException: Unable to connect to the OpenText eDirectory tree.

Solution: The exception might include the name of the object that caused the exception. Verify that the eDirectory tree being imported or deployed to is up and running and that it has OpenText Identity Manager installed.

`The following exception occurred but was not handled. ({0})`

Cause: An unexpected error occurred while resolving an object in eDirectory.

Solution: Contact OpenText Support.

OpenText eDirectory Object/Attribute Creation Errors

`The driver could not be created.`

Cause: Designer attempted to create a driver in OpenText eDirectory, but the process failed.

Solution: Verify that the target OpenText eDirectory server has OpenText Identity Manager installed.

`A [ObjectClass] object named [ObjectName] could not be created.`

Cause: Designer attempted to create a Publisher, Subscriber, or Policy object in eDirectory, but the process failed.

Solution: Verify that the target eDirectory server has OpenText Identity Manager installed.

`The driver password could not be saved.`

Cause: Designer attempted to set the Driver password in OpenText eDirectory, but the request failed.

Solution: Verify that the target eDirectory server has OpenText Identity Manager installed.

`The password named ''{0}'' could not be saved.`

Cause: Designer attempted to set a named password in eDirectory, but the request failed.

Solution: Turn on stack tracing for the Import/Deploy plug-in to get details of the exception. To do this, select **Window > Preferences > Designer for IDM Trace > Enable Tracing**. In the Trace window, select the check box for **Include Stack Traces**.

`The value for the attribute named [Attribute Name] could not be stored on the object named [Object name].`

Cause: Designer attempted to add an attribute to an object in OpenText eDirectory, but the request failed. The error message should contain information about the attribute and object.

Solution: Verify that the attribute and value are valid for the given OpenText eDirectory object type.

`The value for the attribute named ''{0}'' could not be updated using the XSLT on the object named ''{1}''.`

Cause: Unable to export shim configuration information.

Solution: Contact OpenText Support.

`An exception was thrown updating the value of the [Attribute Name] attribute on the [Item Type] object named [Object Name]. [Exception Info]`

Cause: Unable to deploy the OpenText Identity Manager object and attributes to eDirectory. The error message should contain details of the exception.

Solution: Contact OpenText Support.

`A [Object Class] object could not be created. The name is missing.`

Cause: An OpenText eDirectory object could not be created for the given object class because a name was not provided.

Solution: Contact OpenText Support.

`The policy named [Policy Name] contains a cycle in its next transformation list.`

Cause: This is a warning message generated when Designer encounters a circular loop in the policy chain.

Solution: Remove the policy loop by correcting the next policy in the Policy Set view.

`The policies named [Policy name] contain cycles in their next transformation lists.`

Cause: This is a warning message generated when Designer encounters a circular loop in the policy chain.

Solution: Remove the policy loop by correcting the next policy in the Policy Set view.

`Driver [Driver name] could not be restarted for the deployed changes to be in effect.`

Cause: Designer was unable to restart a driver after a deployment.

Solution: Turn on DSTrace screen in OpenText eDirectory to identity the error preventing the driver from starting.

`Driver '[Driver Name]' is disabled and could not be restarted for the deployed changes to be in effect.`

Cause: Designer was unable to restart a driver after a deployment because its Driver Start option is set to **Disabled**.

Solution: Change the Driver Start option to **Manual** or **Auto-start** under the driver properties and then deploy the driver.

`Driver '[Driver Name]' could not be stopped for the deployed changes to be in effect.`

Cause: Designer was unable to stop a running driver after a deployment.

Solution: Turn on DSTrace screen in eDirectory to identify the error preventing the driver from stopping.

`An invalid request to set up security on an exported driver was made, no Driver objects were provided. The request cannot be processed.`

Cause: The code to set up the security equivalence for a deployed driver was passed an invalid parameter.

Solution: Contact OpenText Support.

Warnings

`The version of Identity Manager running on the server named '[Server Name]' does not support all the features of Designer. Although you can import a configuration from that server, changes may not work if the configuration is deployed back to it.`

Cause: An import or deploy action was made to an OpenText eDirectory server running an unsupported version.

Solution: The server must be upgraded for deployments.

`An internal error has occurred. The parameters passed into the importer were invalid.`

Cause: The code that performs the import was passed an invalid parameter.

Solution: Contact OpenText Support.

`The '[Attribute Name]' attribute of '[Object Name]' refers to a policy that does not exist or cannot be accessed.`

Cause: The driver configuration file being processed contains a DN attribute that cannot be resolved in eDirectory.

Solution: Verify or correct the DN attribute value on the specified object in eDirectory.

`An external reference to '[Object Name]' was not handled.`

Cause: The driver configuration file being processed contains a DN attribute that cannot be resolved in eDirectory.

Solution: Contact OpenText Support.

`The XML for the policy named '[Object Name]' contained in the [Policy Type] named '[Policy Name]' does not contain valid XML for a policy. '[Root Node]' is not recognized as the root node for policy XML.`
`The policy is being ignored.`

Cause: The policy being imported does not contain a valid XML document.

Solution: Correct the content of the policy in eDirectory.

```
A [Item Type] can only be imported into a [Item Type].
A [Item Type] can only be imported into a [Item Type] or [Item Type].
```

Cause: An attempt was made to import an OpenText Identity Manager object into an invalid parent object. For example, policies might not be imported into a Driver Set. The code should prevent this from happening, but this error identifies scenarios that were not caught.

Solution: Contact OpenText Support.

```
An unhandled import request was encountered in DeployImporter_Import method
[Object DN].
```

Cause: An attempt was made to import an unknown object or attribute from eDirectory. The code should prevent this from happening, but this error identifies scenarios that were not caught.

Solution: Contact OpenText Support.

```
Could not access the driver configuration file named '[File Name]'.
```

Cause: Designer could not open or parse the given driver configuration file.

Solution: Contact OpenText Support.

```
The driver filter could not be read from the driver named '[Driver Name].
```

Cause: Designer could not import the Driver filter.

Solution: Turn on the DSTrace in eDirectory to determine the error, then contact OpenText Support.

```
An error was encountered processing the driver configuration file. The
variable named [Variable Name] is defined more than once.
```

Cause: The driver configuration file has a variable that is being defined multiple times.

Solution: If you are importing a driver configuration file from a file, edit the file and remove multiple declarations for the specified variable. If this is a dynamically generated configuration file (import/deploy to eDirectory), turn on XML tracing for import/deploy to get a trace of the generated configuration file, then contact OpenText Support. To turn on trace for Designer, select **Window > Preferences > Identity Manager > Application > Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**.

```
An error was encountered processing the driver configuration file. The
declaration of the Node variable named [Variable Name] is invalid. The
[Attribute name] attribute is missing.
```

Cause: The driver configuration file being processed has an invalid variable declaration.

Solution: If you are importing a driver configuration file from a file, edit the driver configuration file and correct the variable declaration. If this is a dynamically generated configuration file (import/deploy to eDirectory), turn on XML tracing for import/deploy to get a trace of the generated configuration file, then contact OpenText Support. To turn on trace for Designer, select **Window > Preferences > Identity Manager > Application > Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**.

`An error was encountered processing the driver configuration file. Flexible prompting requires a 'use-when-value' when a 'use-when-var' is specified.`

Cause: The driver configuration file being processed has an error.

Solution: If you are importing a driver configuration file from a file, edit the driver configuration file and add a use-when-value for the specified use-when-var. If this is a dynamically generated configuration file (import/deploy to eDirectory), turn on XML tracing for import/deploy to get a trace of the generated configuration file, then contact OpenText Support. To turn on trace for Designer, select **Window > Preferences > Identity Manager > Application > Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**.

`An error was encountered processing the driver configuration file. Flexible prompting requires a 'use-when-var' when a 'use-when-value' is specified.`

Cause: The driver configuration file being processed has an error.

Solution: If you are importing a driver configuration file from a file, edit the file and add a use-when-var for the specified use-when-value. If this is a dynamically generated configuration file (import/deploy to eDirectory), turn on XML tracing for import/deploy to get a trace of the generated configuration file, then contact OpenText Support. To turn on trace for Designer, select **Window > Preferences > Identity Manager > Application > Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**.

`The variable named [Variable Name] has been referred to but not defined in the driver configuration file being processed.`

Cause: The driver configuration file has a variable that is being referenced but has not been defined.

Solution: If you are importing a driver configuration file from a file, edit the driver configuration file and add a declaration for the specified variable. If this is a dynamically generated configuration file (import/deploy to eDirectory), turn on XML tracing for import/deploy to get a trace of the generated configuration file, then contact OpenText Support. To turn on trace for Designer, select **Window > Preferences > Identity Manager > Application > Trace > Enable Tracing**. In the Trace window, select the check box for **Include XML Processor Traces**.

`An error was encountered processing the driver configuration file. Built-in variables cannot be used as a flexible prompting control variable. The reference to the variable named '[Variable Name]' is invalid.`

Cause: The driver configuration file being processed contains an invalid reference to a variable.

Solution: If this is a dynamically created configuration file generated during an import/deploy action, contact OpenText Support. If this is a driver configuration file being imported from disk, edit and correct the configuration file for the variable specified.

`An error was encountered processing the driver configuration file. There was a non-checkbox reference to the checkbox variable named '[Check Box Variable name]'.`

Cause: The driver configuration file being processed contains an invalid reference to a check box variable.

Solution: If this is a dynamically created configuration file that is generated during an import/deploy action, contact OpenText Support. If this is a driver configuration file being imported from disk, edit and correct the configuration file for the check box variable specified.

`An error was encountered processing the driver configuration file. An unhandled import prompt was encountered.`

Cause: The driver configuration file being processed contains an invalid prompt type.

Solution: If this is a dynamically created configuration file that is generated during an import/deploy action, contact OpenText Support. If this is a driver configuration file being imported from disk, edit and correct the configuration file.

`The eDirectory tree corresponding to the Identity Vault named '[Identity Vault Name]' cannot be accessed. Directory browsing cannot be performed.`

Cause: Designer attempted to access eDirectory through an eDirectory browse icon in the Driver Configuration Wizard, but the connection could not be created.

Solution: Cancel out of the Driver Configuration Wizard, set up the connection parameters in Identity Vault, and run the Driver Configuration Wizard again.

`The partition could not be created on the ''{0}'' object. The problem may be that it has not replicated to the master yet. You can try creating the partition manually later.`

Cause: Designer attempted to create a partition when deploying a driver set and the partition operation failed.

Solution: Turn on the eDirectory tracing options for partitioning to determine why the eDirectory partitioning operation failed.

`The Driver Set was created but did not replicate to all the servers in the replica ring. The deployment cannot proceed.`

Cause: Designer cannot deploy per-server attributes until the driver set has replicated to the eDirectory server.

Solution: Turn on the eDirectory tracing options for replication and determine why eDirectory replication is not occurring.

`There are no servers associated with the Driver Set named ''{0}''. There must be at least one server associated with any Driver Set being deployed or the Driver Set containing any objects being deployed.`

Cause: Designer cannot deploy an Identity Vault or driver set with an empty server list.

Solution: Edit the properties of the Identity Vault and the driver set to add a server to the server lists.

`The Identity Vault name '[Identity Vault Name]'' does not contain any Driver Set objects to deploy.`

Cause: You cannot deploy an Identity Vault that does not contain at least one driver set.

Solution: Add a driver set to the Identity Vault.

`'[User Name]' could not be authenticated to '[Host Name]'. Cannot proceed.`

Cause: Designer could not authenticate to the eDirectory tree.

Solution: Verify that the hostname, user, and password for the Identity Vault are correct in the Identity Vault properties.

```
The Identity Vault named '[Identity Vault Name]' does not contain the
eDirectory tree to access. Cannot proceed.
```

Cause: The Identity Vault does not contain a host address or DNS name for authentication.

Solution: Specify the host address or DNS name for the Identity Vault in the Properties view or Properties page.

```
Deploy_Util_NoIdentityVault=The {2} named ''{1}'' is not contained in an
{0}. Cannot proceed.
The Identity Vault named '[Identity Vault name]' does not contain the DN of
the user to authenticate to the target eDirectory tree with. Cannot
proceed.
```

Cause: The Identity Vault does not contain a user for authentication.

Solution: Specify the user for the Identity Vault in the Properties view or Properties page.

```
The server list on the parent Driver Set for the following eDirectory
Driver is empty. We were unable to import the connected eDirectory Driver:
```

Cause: Designer uses the per-server Shim Auth Server attribute of an eDirectory driver to identify the tree and connected eDirectory driver to import. Because the server list is empty, the connected eDirectory driver cannot be imported.

Solution: Fix the server list on the driver set for the eDirectory driver and the Drivers Shim Auth Server attribute in eDirectory, or import the connected eDirectory driver separately.

```
The Shim Auth Server parameter for the eDirectory Driver '[Driver Name]' on
server '[Server Name]' is empty. We were unable to import the connected
eDirectory Driver.
```

Cause: Designer uses the Shim Auth Server parameter of an eDirectory driver to identify the tree and connected eDirectory driver to import. If this parameter is empty, the connected eDirectory driver cannot be imported.

Solution: Fix the Shim Auth Server parameter on the eDirectory driver or import the connected eDirectory driver separately.

```
Unable to save Driver Configuration to file '[File Name]'.
```

Cause: Designer was unable to save an exported driver configuration file.

Solution: Try to save the file to a different directory or filename.

```
Unable to clear contents of Driver Configuration file '[File Name]'.
```

Cause: Designer was unable to clear the contents of a driver configuration file that is being overwritten.

Solution: Delete the configuration file being overwritten.

```
Setting up the Security Equals and Excluded objects may only be performed
on a Driver object.
```

Cause: An invalid object was selected in the Modeler or Outline view.

Solution: Select a single Driver object to set up security equivalences or excluded users.

`The selected Driver ''{0}'' has not been deployed or cannot be found in the eDirectory ''{1}''.`

Cause: Designer cannot resolve to the Driver object in eDirectory to set up the security equivalences or excluded user list.

Solution: Deploy the driver to eDirectory before setting up the security equivalences or excluded users.

`The eDirectory tree corresponding to the Identity Vault named '[Tree Name]' cannot be accessed. Setting up the Driver Security Equivalence/Excluded Users cannot be performed.`

Cause: Designer cannot connect or authenticate to the eDirectory tree to set up a driver's security equivalences or excluded user list.

Solution: Verify that the eDirectory parameters specified on the Identity Vault are correct and that the eDirectory server is running.

`The Identity Vault named '[Identity Vault Name]' has no deployment DN specified. It is not deployable.`

Cause: A deployment context is not specified on the Identity Vault or driver set being deployed.

Solution: Add a deploy DN (context) to the properties of the Identity Vault or Driver Set object in Designer.

Reporting Bugs and Giving Feedback

Gathering bugs and getting your ideas are keys to improving the performance of Designer and making Designer a tool of choice for you. To send us your feedback, select **Help** > **Report a Bug or Feedback**. We encourage you to try it!

1. Select **Help** > **Report a Bug or Give Feedback**.
2. Log in to Bugzilla.

   If you don’t have an account, you can easily create one.
3. Select the component in the product that you are reporting on.

   The Designer 4.7 product is preselected. If you don’t know which component you are reporting on, select your best guess (for example, Modeler).
4. In the **Summary** field, summarize the problem or your request for an enhancement.
5. In the **Description** field, describe the bug or enhancement.

   If you are reporting a bug, provide clear steps on how to reproduce the problem.

Designer Does Not Deploy the Global Configuration Attribute on a Driver or a Driver Set

While deploying or reconciling a Global Configuration Value (GCV), Designer only deploys the GCV. It fails to deploy reference attributes responsible for maintaining the linkage.

To workaround this issue, perform the following actions:

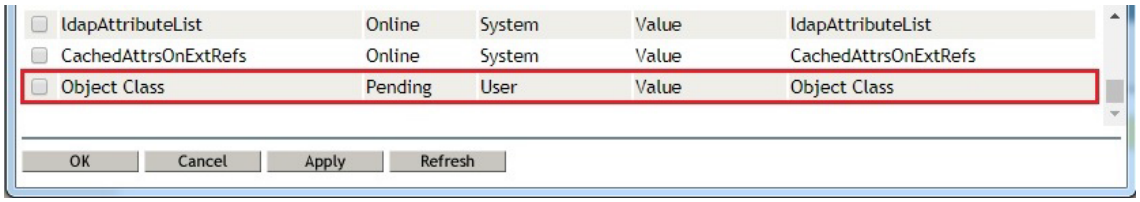
1. Right-click the driver set and select **Live** > **Driver Set Configuration** > **Compare Attributes**.
2. Ensure that only the `DirXML-Globalconfig` attribute is shown and the newly added GCV reference is shown as the difference.
3. Select the driver set in the Compare window.
4. Reconcile the driver set with the Identity Vault.

Designer Takes Too Long To Deploy a Driver or a Driver Set

Sometimes a driver or a driver set takes too long to deploy and Designer appears to hang.

To workaround this issue, add an index for the object class attribute by performing the following actions:

1. In iManager, click **Roles and Tasks > eDirectory Maintenance > Index Management**.
2. Select a server from the list of available servers.
3. On the Modify Indexes page, select the object class attribute for which you want to add an index and select the **Change State** button.



4. Click **OK** to update the index table.
5. Click **Apply** for the changes to take effect.

Manually Removing Invalid Designer Shortcut from mac Launchpad

If you wish to uninstall Designer that you installed from `Identity_Manager_4.7_MacOSX_Designer.tar.gz`, the uninstallation process may not remove the Designer shortcut from Launchpad. To remove the shortcut, reset Launchpad by following the Mac documentation. We recommend that you familiarize with the implications of clearing the Launchpad items before performing this action.

Unable to Launch Designer Application on Mac

While downloading Designer on Mac, sometimes quarantine attribute such as `com.apple.quarantine` is included in the Designer application that prevents you from launching Designer. You must clear the quarantine attributes to solve this issue. To clear the quarantine, open the terminal on Mac and execute the command:

```
# xattr <path where Designer.app is installed>
```

```
# spctl --assess --verbose=4 --type execute <path where Designer.app is installed>
```

```
# xattr -dr com.apple.quarantine <path where Designer.app is installed>
```

NOTE: You will encounter this issue on Mac for any application that has not been downloaded from Mac App Store. This is to ensure that only trusted software runs on your computer.

Designer Fails to Deploy Driversets of Mixed Versions of OpenText Identity Manager

If there are two OpenText Identity Manager servers (for example, two Identity Manager 4.5 servers) in a multi-server environment, and one of them is upgraded to 4.7 (the one that Designer authenticates to), Designer fails to deploy the changes. However, it states that the deployment is successful.

To workaround this issue, perform the following actions:

1. Go to **Driverset Properties > Server** list and remove the OpenText Identity Manager server (In this example, it is the Identity Manager 4.5. server)
2. Deploy the changes
3. Add the OpenText Identity Manager server that you deleted in Step1

Designer Does Not Respond If The Project Name Contains a Space

Designer does not respond if the project name contains a space in it.

To workaround this issue, do not use spaces while naming a project. If your existing project name contains a space, perform the following actions to rename the project:

1. Close all the projects, including the files of projects outside Designer.
2. Go to Project view, and right click on the project name and click **Rename**.
3. Rename the project without any spaces in it and click **OK**.

Designer Fails on Linux to Open Data Item Mapping of a Workflow Activity Containing Non-ASCII Characters

Eclipse Standard Widget Toolkit Table Editor does not support non-ASCII characters. Designer does not properly render the Data Item Mapping view if the workflow activity includes a script containing localized strings with non-ASCII characters.

To resolve this issue, perform the following actions:

1. Create an external ECMAScript object for the workflow activity and add the content of the activity as a function to the ECMAScript object.
2. Open your provisioning request definition in the Provisioning view, right-click it, and then select **Properties**.
3. Add the newly created ECMAScript object as an external script under **Global Scripts**.

   The script is incorporated into the provisioning request definition by the reference using the supplied ECMAScript DN.
4. Click **OK**.
5. Click the Workflow tab to display the workflow.
6. Click the activity, and then click the **Data Item Mapping** tab.
7. Use the newly created script as an inline method in the Data Item Mapping view.

Alternatively, change the localized strings to English.

Importing a Workflow Created Using Older Version of User Application Driver

When you import a workflow that is created using an older version of User Application driver, Designer automatically modifies the workflow request as per the current version of User Application driver. If the User Application driver version is not available, it defaults to the 3.6.1 version of the User Application driver. This behavior of Designer is by design.

Form Builder Does Not Launch Properly From Designer on Linux Platforms

When you launch Form Builder from Designer on Linux, the Form Builder displays a blank page and does not launch properly.

To workaround this issue, increase the memory of the machine where Designer is installed.

Unable to Deploy Large PRDs

While trying to deploy PRDs whose size are greater than 16 MB, the PRD deployment fails with an error message.

Workaround: This is a limitation with JDK. To work around this issue, perform the following steps:

1. Export the PRD from Designer.
2. Create a LDIF file from the PRD.

   NOTE: Ensure that the value of `dn` and `cn` in the LDIF file are the same.

3. Copy the LDIF file to the server where you want to deploy the PRD.
4. Run the following command to deploy the LDIF file:

   ```
   ldapadd -x -h <IP Address>-D "cn=admin,ou=sa,o=system" -w <password> -f
   <name of the LDIF file>
   ```

5. Import the PRD into Designer.
6. To verify the details of the PRD, navigate to **Provisioning** > **User Application driver** > **Provisioning Request Definitions** > **Accounts**.
7. (Conditional) If you want to update the existing PRD, perform the following steps:
   a. Export the PRD from Designer.
   b. Create a LDIF file from the PRD.
   c. Add the following line in the LDIF file:

      ```
      typeChange: modify
      ```

   d. Run the following command to deploy the modified LDIF file:

      ```
      ldapmodify -x -h <IP Address>-D "cn=admin,ou=sa,o=system" -w
      <password> -f <name of the LDIF file>
      ```

Unable to Connect to the Git Repository After Committing Package Changes for the Second Time

If you try to update Designer packages, then Designer displays the **Can’t Connect to any repository** error message in the following scenarios:

- you have set basic authentication for package updates, and
- you also try to perform any Git operations on existing repositories.

The issue is not observed when you perform Git operations on a new repository.

Workaround: Restart Designer after updating Designer packages.

Unable to Launch Form Builder on Linux Platforms

When you launch the 4.8.2 version of Form Builder, the **libXss.so.1: cannot open shared object file: No such file or directory** error message is displayed. This issue is observed on some Linux platforms only.

Workaround: Install the `libXScrnSaver-1.2.2-6.1.el7.x86_64.rpm`.

NOTE: We recommend you to obtain the dependent packages from your operating system subscription service to ensure continued support from your operating system vendor. If you do not have a subscription service, you can find the recent packages from a website such as http://rpmfind.net/linux.

Modifying The src Attribute Used in a Policy

The usage of `src` attribute in the token-map verb of a policy was discontinued with OpenText Identity Manager 4.10. The `src` attribute is replaced with the `source` attribute.

However, if you have a policy that was created with versions prior to OpenText Identity Manager 4.10, and you want to modify all the occurrences of `src` to `source`, try the following workaround.

In the Identity Manager 4.8.3 version, Designer introduces a new **Replace** feature that allows you to search for strings and replace them as per your convenience.

1. Launch Designer.
2. Import the required project to Designer.
3. From the Modeler view, launch the **Search/Replace dialog** box.

   For more information about launching the Search/Replace option, see The Search Results View in the OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts.
4. In the **Search** field, specify the value as `"src="`.
5. Select the **Enable Replace** checkbox.
6. In the **Replace** field, specify the value as `"source="`.
7. From the **Filter** list, select the policy script.
8. Click **OK**.
9. Save and deploy the changes.

Modifying the do-create-resource Action Manually to Use REST API

The `do-create-resource` action is enhanced to use REST API with the OpenText Identity Manager 4.10 version. However, if you are on versions prior to Identity Manager 4.8.2, this can only be achieved by manually editing the XML file.

The following procedure provides instructions on modifying this setting manually through the XML file:

1. Log in to Designer.
2. Open a policy in the policy editor.
3. Select the **create resource** action in the **Do** field.
4. Click the **XML Source** tab.
5. To use REST API, set the value of the **use-rest** parameter to `true` in the XML file. To use SOAP API, set the value to `false`. By default, the value is set to `false`.

This capability is available through the Designer policy builder from the Identity Manager 4.8.2 onwards. For more information, see Create Resource action in the OpenText™ Identity Manager CE 24.4 (v4.10) - Using Designer to Create Policies.

A Modeler Operations

Modeler operations are available when you right-click inside the Modeler. The list of operations depends upon whether you right-click the Modeler space or one of the objects.

- "Modeler Space Operations" on page 561
- "Identity Vault Operations" on page 563
- "Driver Set Operations" on page 566
- "Driver Operations" on page 569
- "Application Operations" on page 573
- "Submenus" on page 577
- "Keyboard Support" on page 578

Modeler Space Operations

The following figure illustrates Modeler options that are available when you right-click empty Modeler space.

Figure A-1 *Modeler-Space Operations*

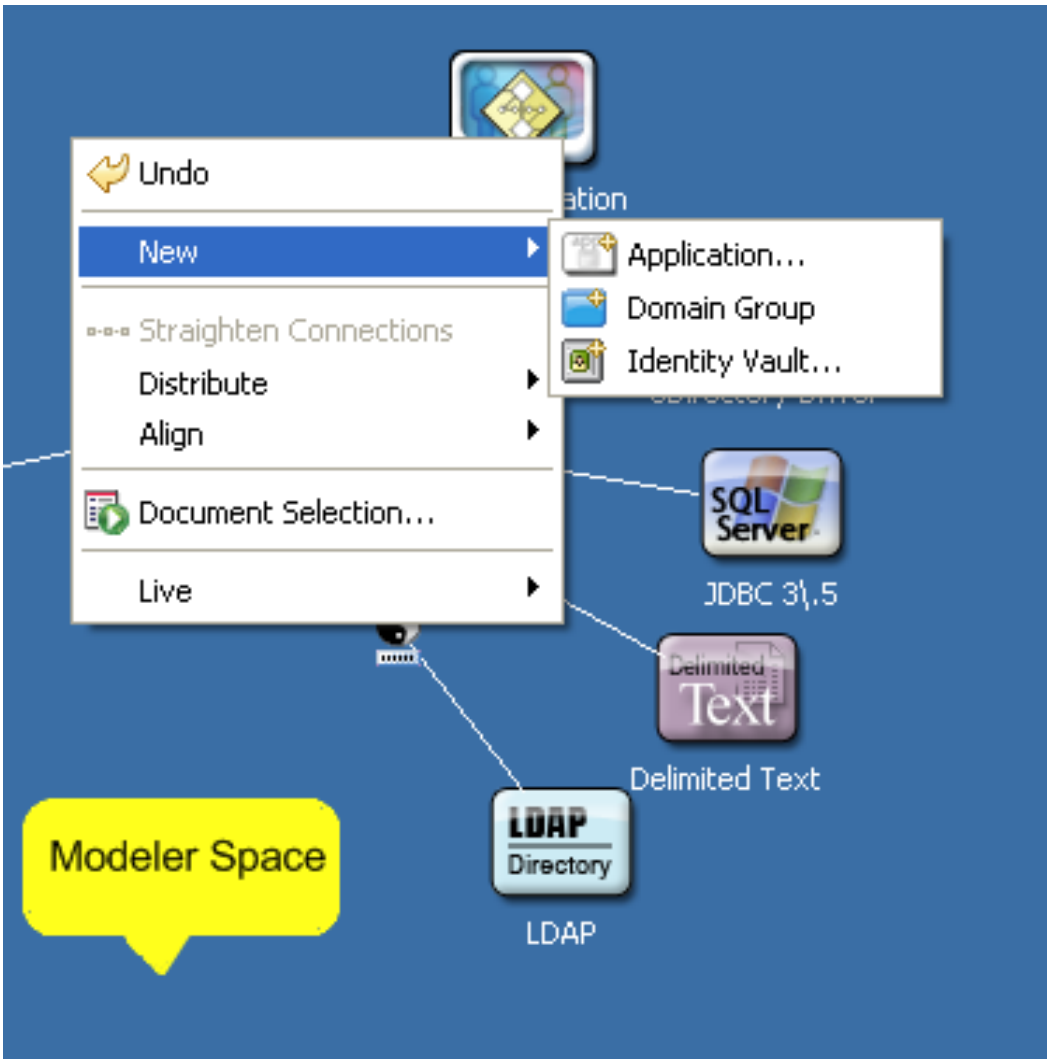


Table A-1 *Modeler-Space Operations*

Right-Click Operation	Description
Undo	Returns an item to its previous status.
New > Application	Selects an application from a list and places the application to the Modeler.
New > Domain Group	Places a Domain Group in the Modeler.
New > Identity Vault	Launches a dialog box that specifies a server and creates an Identity Vault.

Right-Click Operation	Description
Straighten Connections	Straightens lines for selected items. For example, you can straighten a line to a driver, all lines in a driver set, everything in a Domain Group, or an entire project. If a line is not within a few degrees of being horizontal or vertical, this option is dimmed.
Distribute	Evenly distributes applications vertically or horizontally. Press Ctrl, select the items that you want to distribute, then select a pattern.
Align	Aligns applications according to a pattern that you select. Press Ctrl, select the items, then select a pattern (for example, Align Bottom).
Document Selection	Launches the Document Generation Wizard, which documents your project.
Live > Import	Imports an Identity Vault.

Identity Vault Operations

The following figure illustrates Modeler operations that are available when you right-click an Identity Vault.

Figure A-2 *Identity Vault Operations*

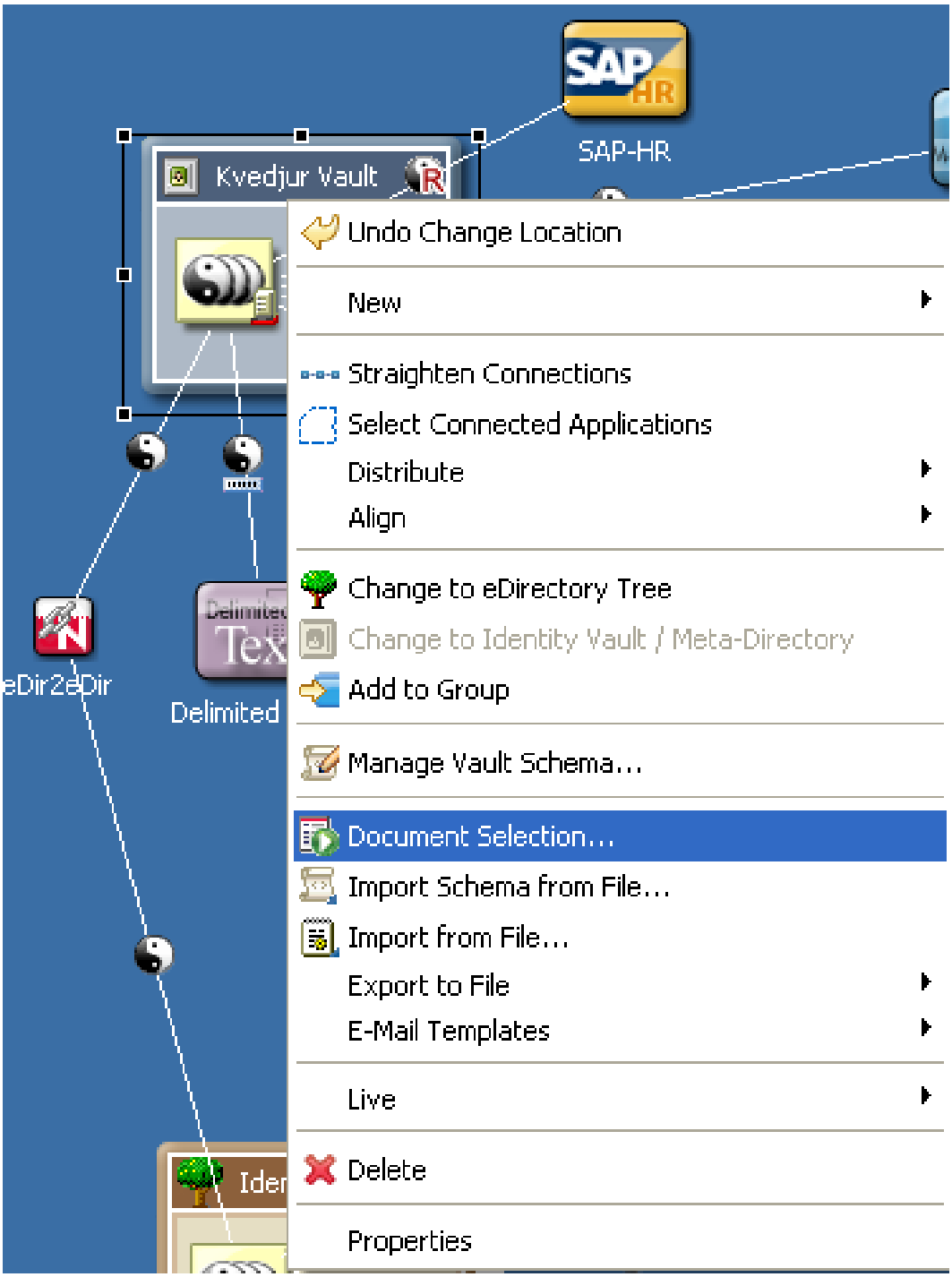


Table A-2 *Identity Vault Operations*

Operation	Description
Undo	Returns an item to its previous status.
New > Driver Set	Adds a Driver Set object to an Identity Vault.

Operation	Description
New > Library	Launches the New Library Wizard.
New > Server	Launches the Add Server Wizard.
Straighten Connections	Straightens lines for selected items. For example, you can straighten a line to a driver, all lines in a driver set, everything in a Domain Group, or an entire project. If a line is not within a few degrees of being horizontal or vertical, this option is dimmed.
Select Connected Applications	Selects all applications that are connected to the driver set or Identity Vault. This is convenient if you have several applications connected to a driver set. You can quickly move them all or delete them all without browsing to and selecting each one.
Distribute	Evenly distributes applications vertically or horizontally. Press Ctrl, select the items that you want to distribute, then select a pattern.
Align	Aligns applications according to a pattern that you select. Press Ctrl, select the items, then select a pattern (for example, Align Bottom). See Table A-6 on page 577.
Change to eDirectory Tree	Changes an Identity Vault to an eDirectory tree. In Architect mode, this option displays a tree instead of a vault. This is just for diagramming purposes; there is no functional difference.
Change to Identity Vault	Changes an eDirectory tree into an Identity Vault. In Developer mode, this option displays a vault instead of a tree. This is just for diagramming purposes; there is no functional difference.
Add to Group	Creates a Domain Group, and adds the selected items to it. The selected items are removed from any group to which they were previously associated.
Manage Vault Schema	Launches the Schema Manage tool, from which you can manipulate schema settings for the selected Identity Vault or directory.
Document Selection	Launches the Document Generation Wizard, which documents the selected Identity Vault.
Import Schema from File	Enables you to browse to a file and import a schema into a `.sch` or `.ldif` file.
Import from Configuration File	Allows you to browse to and import a driver configuration file.
Export to File > Configuration	Exports the Identity Vault to a `.xml` file. iManager can consume this format, and Designer can re-import it. For more information, see "Exporting to a File" on page 401.
Export to File > Schema	Exports the schema to a `.sch` or `.ldif` file.

Operation	Description
E-mail Templates > E-Mail Server Properties	Configures an e-mail server to send e-mail notifications. Edits templates used to notify users concerning password events. For more information, see "Configuring the E-Mail Server" on page 280.
E-Mail Templates > Edit Templates	Launches the E-mail Templates dialog box, from which you can edit the e-mail templates associated with the selected Identity Vault. For more information, see Chapter 10, "Setting Up E-Mail Notification Templates," on page 271.
E-Mail Templates > Update Templates	Adds localized templates to the Default Notification Collection.
Live > Import	Enables you to connect to a server, browse to and select objects, and import the objects into the Identity Vault.
Live > Deploy	Prepares a deployment summary and then deploys selected objects and attributes.
Live > Compare	Compares selected Identity Vaults. Enables you to reconcile or update Identity Vaults. See "Using the Compare Feature When Deploying" on page 394.
Live > Schema > Import	Imports the schema from an existing Identity Vault.
Live > Schema > Deploy	Deploys the modified or imported schema.
Live > iManager	Enables you to connect to a server and launch iManager.
Live > Status for All Drivers	Lists drivers that are stopped or running.
Live > Start All Drivers	Starts all drivers associated with the selected object.
Live > Stop All Drivers	Stops all drivers associated with the selected object.
Live > Restart All Drivers	Restarts all drivers associated with the selected object.
Delete	Deletes the Identity Vault.
Properties	Displays the Identity Vault's properties pages.

Driver Set Operations

The following figure illustrates Modeler operations that are available when you right-click a driver set.

Figure A-3 *Driver Set Operations*

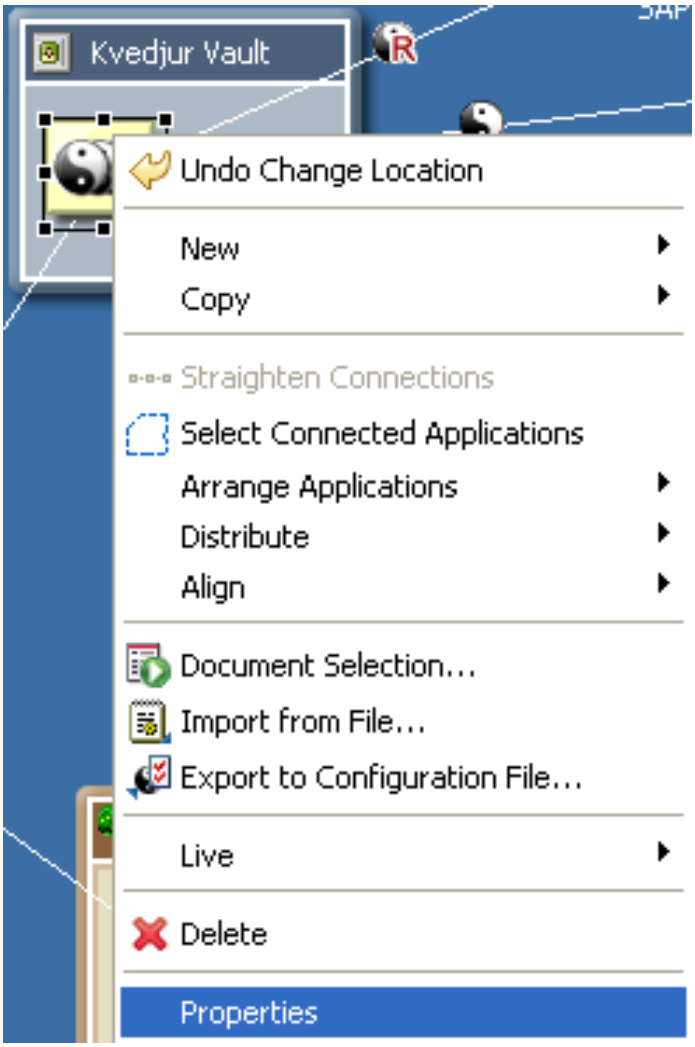


Table A-3 *Driver Set Operations*

Operation	Description
New > Driver	Launches the Driver Configuration Wizard to add a driver to the driver set.
New > Job	Launches the Job Scheduler Wizard to create a job.
New > Library	Launches the New Library Wizard.
New > Role-Based Entitlement Policies	Creates an Entitlement policy that is a dynamic group with additional features added to grant entitlements on connected systems.
New > DS Object	Creates a DS object that is part of packages. A DS object contains information that creates eDirectory objects in the Identity Vault.

Operation	Description
New > Global Configuration	Creates a Resource object that stores global configuration values that can be applied in a package.
Copy > Driver Set Settings	Enables you to browse to a driver set and copy its settings. A pasted copy overwrites data in the target driver set.
Copy > Global Configuration Values	Enables you to copy Global Configuration Values (GCVs) from one driver set to one or more other driver sets. This option enables you to configure GCVs in one place and then apply GCV settings to selected targets.
Straighten Connections	Straightens all lines in the driver set. If a line is not within a few degrees of being horizontal or vertical, this option is dimmed.
Select Connected Applications	Selects all applications that are connected to the driver set. You can quickly move or delete the applications without browsing to and selecting each one.
Arrange Applications	Arranges application icons around their associated driver set icon. A check mark indicates the current layout for the driver set. After the layout is set, any applications that you connect are automatically snapped into that layout. For more information, see Table A-7 on page 577.
Distribute	Evenly distributes applications vertically or horizontally. Press Ctrl, select the items that you want to distribute, then select a pattern.
Align	Aligns applications according to a pattern that you select. Press Ctrl, select the items, then select a pattern (for example, Align Bottom). See Table A-6 on page 577.
Document Selection	Launches the Document Generation Wizard, which documents the selected driver set.
Import from Configuration File	Reads in exports made from iManager or Designer. For more information, see “Importing a Driver Configuration File” on page 302.
Export to Configuration File	Exports the driver set to a `.xml` file. iManager can consume this format, and Designer can re-import it. For more information, see “Exporting to a File” on page 401.
Live > Import	Enables you to connect to a server, browse to and select objects, and import the objects into the driver set.
Live > Deploy	Prepares a deployment summary and then deploys selected objects and attributes.
Live > Compare	Compares selected driver sets. Enables you to reconcile or update driver sets. See “Using the Compare Feature When Deploying” on page 394.
Live > Driver Set Configuration > Import Attributes	Imports attributes from an existing driver set.

Operation	Description
Live > Driver Set Configuration > Deploy Attributes	Deploys the modified or imported attributes.
Live > Driver Set Configuration > Compare Attributes	Compares attributes in Designer to the connected OpenText Identity Manager server.
Live > Status > for All Drivers	Lists drivers that are stopped or running.
Live > Start All Drivers	Starts all drivers associated with the selected object.
Live > Stop All Drivers	Stops all drivers associated with the selected object.
Live > Restart All Drivers	Restarts all drivers associated with the selected object.
Delete	Deletes the driver set.
Properties	Enables you to configure Identity Vaults, driver sets, drivers, and applications.

Driver Operations

The following figure illustrates Modeler operations that are available when you right-click a driver.

Figure A-4 *Driver Operations*

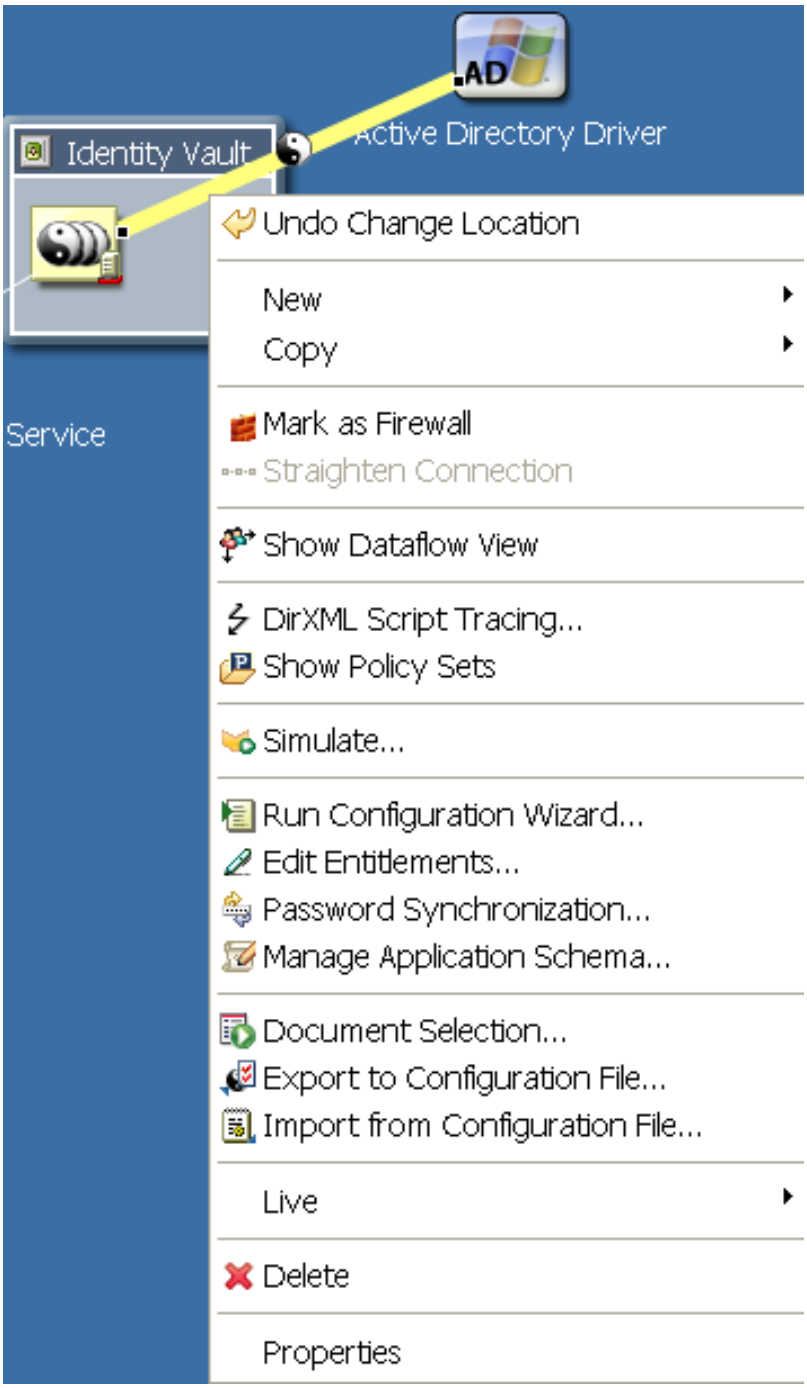


Table A-4 *Driver Operations*

Operation	Description
Undo	Returns an item to its previous status.

Operation	Description
New > Credential Application	Creates a new Application object, which stores static single sign-on parameters for a specific application. For more information, see *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.
New > Credential Repository	Creates a new Repository object, which stores static configuration information for an authentication credential repository such as either SecretStore or SecureLogin. For more information, see the overview section in the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.
New > DirXMLScript	Launches the Policy Builder, creates a policy, and creates a new DirXML Script. DirXML Script is the primary method of implementing policies in the OpenText Identity Manager engine.
New > ECMAScript	Creates an ECMAScript object and opens the ECMAScript editor.
New > Entitlement	Launches the Entitlement Wizard and adds an entitlement to the selected driver. For more information, see Chapter 13, “Using Entitlements,” on page 349.
New > Job	Launches the Job Scheduler Wizard to create a job.
New > Mapping Table	Creates a Mapping Table object. A policy uses a Mapping Table object to map one set of values to another set of corresponding values.
New > Resource	Creates a Resource object. Resource objects (for example, generic, ECMAScript, mapping table, application, or repository resources) store information that drivers use. The information can be arbitrary data in any format (for example, XML or text).
New > DS Object	Creates a DS object that is part of packages. A DS object contains information that creates eDirectory objects in the Identity Vault.
New > Global Configuration	Create a Resource object that stores global configuration values that can be applied in a package.
New > Schema Map	Creates a schema map policy and launches the Schema Map editor. A schema map policy maps class names and attribute names between the Identity Vault namespace and the application namespace. The schema map policy is applied in both directions.
New > XSLT	Creates an XSLT policy. XSLT is a standard language for transforming XML documents. You can use the XSLT option to implement policies as XSLT style sheets.
New > From Copy	Creates a policy by copying from an existing object.
Copy > Settings	Copies data from the selected driver to a target driver. A pasted copy overwrites data in the target driver.

Operation	Description
Copy > Server-Specific Settings	Copies data from the selected server to a target server. A pasted copy overwrites data in the target server.
Mark/Unmark as Firewall	Enables you to mark where a driver is communicating through a firewall. Used in Developer mode. If driver icons are turned off, the firewall icon doesn't appear.
Straighten Connection	Straightens a driver connection line. If a line is not within a few degrees of being horizontal or vertical, this option is dimmed.
Show Dataflow View	Displays the flow of information between the application and the driver in the Developer view. Launches the Dataflow view. For more information, see Chapter 8, "Managing the Flow of Data," on page 245.
Dataflow	Displays the dataflow between the application and driver set. Appears only when dataflow view is activated.
DirXML Script Tracing	Turns on or off the tracing of rules, conditions, condition groups, actions, and tokens at the driver level.
Show Policy Sets	Launches the Policy Set and Policy Flow views. For more information, see "Policy Set View" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.
Simulate	Runs the Simulate Policy Transformation program against the selected driver.
Run Configuration Wizard	Guides you through creating a driver. After you fill in the wizard's forms, Designer automatically generates policies that configure the driver to function as described in the forms.
Edit Entitlements	Enables you to select an entitlement that is associated with the driver and edit the entitlement's settings. For more information, see Chapter 13, "Using Entitlements," on page 349.
Password Synchronization	Configures and displays the flow of password synchronization. For more information, see "Integrating Passwords" on page 265.
Manage Application Schema	Enables you to manage a copy of the managed system's schema. You can make changes to a copy of the application schema so that you can test the OpenText Identity Manager drivers in Designer. See "Managing a Copy of an Application Schema" on page 166.
Document Selection	Launches the Document Generation Wizard, which documents the selected driver.
Export to Configuration File	Exports the driver to a `.xml` file. iManager can consume this format, and Designer can re-import it. For more information, see "Exporting to a File" on page 401.
Import From Configuration File	Imports an exported `.xml` driver file.

Operation	Description
Live > Import	Enables you to connect to a server, browse to and select a driver, and import the objects into the driver.
Live > Deploy	Prepares a deployment summary and then deploys selected objects and attributes.
Live > Compare	Compares selected drivers. Enables you to reconcile or update drivers. See "Using the Compare Feature When Deploying" on page 394.
Live > Driver Configuration > Import Attributes	Imports attributes from an existing driver.
Live > Driver Configuration > Deploy Attributes	Deploys the modified or imported attributes.
Live > Driver Configuration > Compare Attributes	Allows you to compare the attributes of a policy to the attributes that are already deployed.
Live > Refresh Application Schema	Specifies the server on an eDirectory tree where the schema is refreshed after an application's schema changes. See "Refreshing the Application Schema" on page 167.
Live > Status for All Drivers	Reports whether the driver is stopped or running.
Live > Start Driver	Starts the driver.
Live > Stop Driver	Stops the driver.
Live > Set Driver Trace Level	Specifies how much information to display in a trace level log from the driver. Settings go from 0-5.
Live > Restart Driver	Restarts the drivers.
Live > Set Up Driver Security	Launches the Driver Security Equals/Exclusions dialog box. Enables you to configure the selected driver's security equivalences and to exclude selected users from administrative roles. If you select multiple drivers, this dialog box lets you add, modify, and remove common security equivalences and exclusions of the selected drivers.
Delete	Deletes the selected driver and its policies.
Properties	Launches the driver's property pages. Enables you to configure the driver.

Application Operations

The following figure illustrates Modeler operations that are available when you right-click an application.

Figure A-5 *Application Operations*

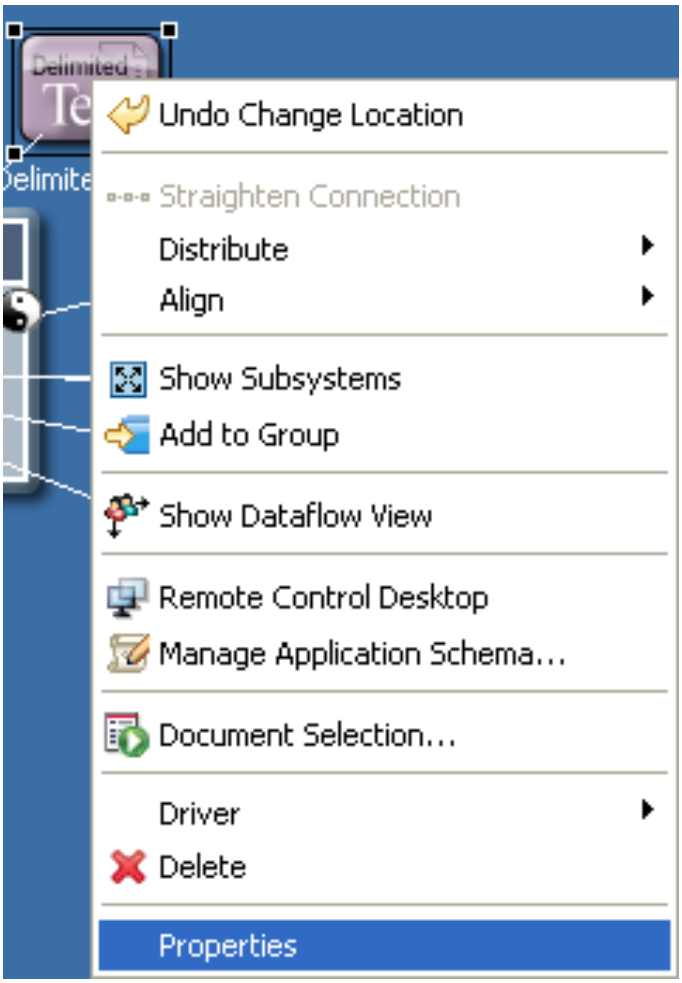


Table A-5 *Application Operations*

Operation	Description
Undo Change Location	Returns an item to its previous status.
Disconnect eDir-to-eDir	(Viewable when you select an eDir-to-eDir application) Separates the eDir-to-eDir application into two eDirectory drivers.
Straighten Connection	Straightens a driver connection line. If a line is not within a few degrees of being horizontal or vertical, this option is dimmed.
Distribute	Evenly distributes applications vertically or horizontally. Press Ctrl, select the items that you want to distribute, then select a pattern.
Align	Aligns the selected objects horizontally and vertically. For more information, see Table A-6 on page 577.

Operation	Description
Change to eDirectory Tree	(Viewable when you select an eDirectory application.) Runs the Driver Configuration Wizard to install an eDir-to-eDir driver. Places a tree icon in the Identity Vault.
Change to Identity Vault/Meta-Directory	(Viewable when you select an eDirectory application.) Runs the Driver Configuration Wizard to install an eDir-to-eDir driver. Places a vault con in the Identity Vault.
Show/Hide Subsystems	Lets you model an application's or operating system's subsystems. For example, if you have a Linux system, you can open it and drop MySQL inside as a subapplication that runs on Linux. This is for diagramming purposes only, but can be convenient for accurately capturing the structure of the enterprise systems around which you are building the identity solution.
Add to Group	Creates a Domain Group, and adds the selected items to it. The selected items are removed from any group that they were previously associated with.
Show Dataflow View	Displays the flow of information between the application and the driver in the Developer view. Launches the Dataflow view and lists Dataflow on the menu. For more information, see Chapter 8, "Managing the Flow of Data," on page 245.
Remote Control Desktop	Launches a remote control session for the selected application. The host server must have an existing VNC server running.
Manage Application Schema	Enables you to manage a copy of the managed system's schema. You can make changes to a copy of the application schema so that you can test the OpenText Identity Manager drivers in Designer. See "Managing a Copy of an Application Schema" on page 166.
Document Selection	Launches the Document Generation Wizard, which documents the application.
Driver > **DirXML Script Tracing**	Turns on or off the tracing of rules, conditions, condition groups, actions, and tokens at the driver level.
Driver > Show Policy Sets	Launches the Policy Set and Policy Flow views. For more information, see "Policy Set View" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Designer Concepts*.
Driver > Simulate	Runs the Simulate Policy Transformation program against the selected driver.
Driver > Run Configuration Wizard	Guides you through creating a driver. After you fill in the wizard's forms, Designer automatically generates policies that configure the driver to function as described in the forms.
Driver > Password Synchronization	Configures and displays the flow of password synchronization. For more information, see "Integrating Passwords" on page 265.
Driver > Document Selection	Launches the Document Generation Wizard, which documents the driver.

Operation	Description
Driver > Export to Configuration File	Exports the driver to a `.xml` file. iManager can consume this format, and Designer can re-import it. For more information, see "Exporting to a File" on page 401.
Driver > Import from Configuration File	Allows you to browse to and import a driver configuration file
Driver > Import	Enables you to connect to a server, browse to and select a driver, and import the objects into the driver.
Driver > Deploy	Prepares a deployment summary and then deploys selected objects and attributes.
Driver > Compare	Allows you to compare the information structure in Designer on an object, to the object that is deployed or running on an eDirectory server.
Driver > Driver Configuration > Import Attributes	Imports attributes from an existing driver.
Driver > Driver Configuration > Deploy Attributes	Deploys the modified or imported attributes.
Driver > Driver Configuration > Compare Attributes	Allows you to compare the attributes of a policy to the attributes that are already deployed.
Driver > Status for All Drivers	Reports whether the drivers are stopped or running.
Driver > Start Driver	Starts the driver.
Driver > Stop Driver	Stops the driver.
Driver > Set Driver Trace Level	Allows you to specify how much information you want to see in a trace level log from the driver. Settings go from 0-5.
Driver > Restart Driver	Restarts the drivers.
Driver > Set Up Driver Security	Launches the Driver Security Equals/Exclusions dialog box. Enables you to configure the selected driver's security equivalences and to exclude selected users from administrative roles. If you select multiple drivers, this dialog box lets you add, modify, and remove common security equivalences and exclusions of the selected drivers.
Driver > Properties	Launches the driver's property pages. Enables you to configure the driver.
Delete	Deletes the application and driver.
Properties	Enables you to configure Identity Vaults, driver sets, drivers, and applications.

Submenus

Table A-6 *Align Submenu*

Operation	Description
Align Top	Aligns the top edge of the selected objects.
Align Bottom	Aligns the bottom edge of the selected objects.
Align Left	Aligns the left edge of the selected objects.
Align Right	Aligns the right edge of the selected objects.
Align Center	Horizontally aligns the centers of the selected objects.
Align Middle	Vertically aligns the middles of the selected objects.

Table A-7 *Arrange Applications Submenu*

Operation	Description
Arrangement Off	Disables a previously selected auto-arrangement method.
Box	Arranges application icons in a square around the driver set.
Circle	Arranges application icons in a circle around the driver set.
Half Circle	Arranges application icons in a semicircle around the driver set.
Star	Arranges application icons in a star around the driver set.
Fan Out - Bottom	Arranges application icons in a fan shape below the driver set.
Fan Out - Left	Arranges application icons in a fan shape to the left of the driver set.
Fan Out - Right	Arranges application icons in a fan shape to the right of the driver set.
Fan Out - Top	Arranges application icons in a fan shape above the driver set.
Expand/Contract	Expands or contracts the layout of the application icons. Selecting this option opens a dialog box from which you drag the slide in the **Factor** field to change the layout.

Table A-8 *Dataflow Submenu*

Operation	Description
Publish	Specifies that the Publisher channel is synchronized for the selected objects (uni directional sync from selected objects.) For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.
Subscribe	Specifies that the Subscriber channel is synchronized on the selected objects (unidirectional sync to selected objects.) For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.
Sync	Specifies that both the Publisher and Subscriber channel are synchronized for the selected objects (bidirectional sync.) For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.
Ignore from	Specifies that the selected objects ignore Subscriber channel synchronization. For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.
Ignore to	Specifies that the selected objects ignore both Publisher and Subscriber channel synchronization. For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.
Ignore both directions	Specifies that the selected objects ignore Publisher channel synchronization. For more information, see Chapter 8, “Managing the Flow of Data,” on page 245.

Table A-9 *Distribute Operations Submenu*

Operation	Description
Horizontal	Evenly spaces the selected objects horizontally.
Vertical	Evenly spaces the selected objects vertically.

Keyboard Support

The following table describes common keyboard shortcuts available in the Modeler.

Table A-10 *Shortcut Keys*

Keystroke	Description
/	Navigates to the item's next connection.

Keystroke	Description
\	Navigates to the item's previous connection.
Delete	Deletes the selected item or line.
Left-arrow	Navigates left.
Right-arrow	Navigates right.
Up-arrow	Navigates up.
Down-arrow	Navigates down.
<Alt>+Down-arrow	Navigates into a subgroup.
<Alt>+Up-arrow	Navigates out of a subgroup.
<Ctrl> + =	Zooms in.
<Ctrl> + -	Zooms out.
<Ctrl> + A	Selects all objects in the current project.
<Ctrl> + C	Copies the selected objects to the Clipboard.
<Ctrl> + F	Opens the Find dialog box for searching the project.
<Ctrl> + V	Pastes the Clipboard contents to the selected location.

B Document Generator Core Support Templates

This is reference information to help you customize the Document Generate feature.

- “dgSection.xsl” on page 581
- “dgFormat.xsl” on page 582
- “idmConfig.xsl” on page 585
- “idmUtil.xsl” on page 586

dgSection.xsl

Template	Description
match “/”	Main template that invokes all sub-templates. Users can override this template to create their own template behavior; however, you should override the Section.Content, Section.Body, or Section.Main.
Section.Sequence	Main template that invokes all sub-templates. Users can override this template to create their own template behavior; however, you should override the Section.Content, Section.Body, or Section.Main.
Section.Main	This section includes Section.Content and Section.Children.
Section.Content	This section includes Section.Title and Section.Body.
Section.Body	The body content of the section.
Section.ShowStyleAttributes	Describes the default way to display attributes when no template is defined. Parameters: ◆ border - border used for tables. The default value is 0.5pt solid black.
Section.Children	Inserts the child sections that are passed as a parameter into this template.
Section.PageLayout	Formats the page layout, including paper size, headers, page numbering, and so forth. The Section.Main template is called to insert the document into this layout.
Section.staticContent	Formats the page layout, including paper size, headers, page numbering, and so forth. The Section.Main template is called to insert the document into this layout.

Template	Description
Section.Title	Creates a title block containing the appropriate title text and link.
Section.TitleText	Gets the text to be displayed for this section.

dgFormat.xsl

Template	Description
Format.Title	This template handles all of the details involved in formatting a title block. Parameters: ◆ text - Text to display. ◆ id - id for linking to this title (such as from the table of contents). ◆ font - Font size to use. ◆ image - Image to show as a bullet. The auto value tries to determine an image based on the current element.
Format.FigureTitle	This template handles all of the details involved in formatting a figure title block. Parameters: ◆ title - Title text. ◆ description - Description text.
Format.OutputTextArea	Formats parameter information returned from a text area control that can contain HTML tags. If there is no HTML prefix, line breaks are inserted. Parameters: ◆ value - Value of the textarea to output.
Format.EnabledStatus	Shows the enabled image if the value is True. The disabled image shows only when the showDisabled parameter is set to True. Parameters: ◆ value - Enabled, True/False. ◆ showDisabled - Set to True if the disabled image should show when the value is False. The default value is False.

Template	Description
Format.Chechbox	Shows a check box image if the value is True, or an empty check box image otherwise. Parameters: ◆ value - Checked, True/False. ◆ default - Default value if “”, False, or some other value other than True exists. The default value is False.
Format.PropertyRow	Shows a table property row with two columns, one for the name and one for the value. Parameters: ◆ propertyName - The property name. ◆ propertyValue - The property value. ◆ border - Border used for the table. The default value is 0.5pt solid black. ◆ disable-output-escaping - Disables output escaping on the output value, so you can pass escaped FO content. The default value is False. ◆ showEmpty - Show empty values. The default value is False.
Format.ContextRow	Shows a contextual row with related attribute. Use this inside a table. Parameters: ◆ text - Text to display. ◆ level - Level or indent. The default value is 1. ◆ href - HREF value to link to another portion of the document. ◆ image - Image to show as a bullet. The auto value tries to determine an image based on the current element. ◆ show-page-ref - Show page reference; True/False. The default value is True.
Format.ShowBulletImage	Show a bullet image. Parameters: ◆ image - Image to show as a bullet. The auto value tries to determine an image based on the current element.

Template	Description
Format.XMLFigure	This template takes care of all the details involved in formatting a figure that shows XML content. Parameters: ◆ title - Title text. ◆ description - Description text. ◆ xml - XML data to show in the figure in text. You can also use a "." to get the current node and children. ◆ simple-format - If True, this shows the XML without text selecting. This can also be preferred if name space attributes need to be included or if the XML is not well-formed. The default value is False.
match "node()" mode "xml-to-text"	XML-to-text formatting function. Parameters: ◆ attr-name-color, attr-value-color
match "@*" mode "xml-to-text"	XML-to-text formatting function. Parameters: ◆ attr-name-color, attr-value-color
match "text()" mode "xml-to-text"	XML-to-text formatting function.
match "comment()" mode "xml-to-text"	XML-to-text formatting function. Parameters: ◆ comment-color
Format.ImageFigure	Formats a figure that shows an image for its content. Parameters: ◆ title - Title text. ◆ description - Description text.
Format.PageBreak	Inserts a page break.
Format.BasicLink	Creates a basic link to the given HREF using the given text. If the `href` parameter is empty, it only outputs the text value. Parameters: ◆ text - Link text ◆ href - Link HREF

Template	Description
Format.BasicLinkToReferencedItem	Creates a basic link to the XSI referenced item. This uses the @guid attribute to build the link. If no @guid is available, only the text label is rendered. Parameters: ◆ xsiHref - XSI value of referenced node.
Format.Uppercase	Used to convert a string to uppercase text. Parameters: ◆ value - The value you want to convert to uppercase.
Format.SmartSpace	Used to convert a string to smart-spaced text. Parameters: ◆ value - The value you want to smart-space.
Format.OutputDebugParameters	Outputs the debug parameters for a section when the DEBUG_PARAMS attribute is enabled.
Format.Debug	Outputs the specified text in a debug format when the DEBUG attribute is enabled. Parameters: ◆ text - Debug text.

idmConfig.xsl

Template	Description
match "*" mode "xmlFigure"	Build an XML Figure for any policy type. Parameters: ◆ title, description, alwaysShowPolicyXmlSource
match "xsl:stylesheet|xsl:transform"	Filter. Parameters: ◆ title, description
match "attr-name-map"	Attribute mapping.
match "policy"	Policy matching. Parameters: ◆ title, description, alwaysShowPolicyXmlSource
opConcat	

Template	Description
opDelim	
match "@*" mode "DirXMLScript"	
match "*" mode "DirXMLScript"	
match "arg-actions" mode "DirXMLScript"	
match "arg-dn" mode "DirXMLScript"	
match "arg-value" mode "DirXMLScript"	
match "token-text" mode "DirXMLScript"	
getLabel	Utility method used to get policy related text labels. Parameters: ◆ name - The name of the label.
match "actions"	

idmUtil.xsl

Template	Description
IdmUtil.ItemPropertyTable	Shows a table of values for the current Item. Depending on the item, it might filter attributes. Parameters: ◆ title - Title text. ◆ description - Description text. ◆ showEmpty - Show empty values. The default value is False.
IdmUtil.StartOptionPropertyRow	Shows the appropriate icon and text for the startup option on the current Item. (0 = Disabled, 1 = Manual, 2 = Auto) Parameters: ◆ propertyName, propertyValue, border

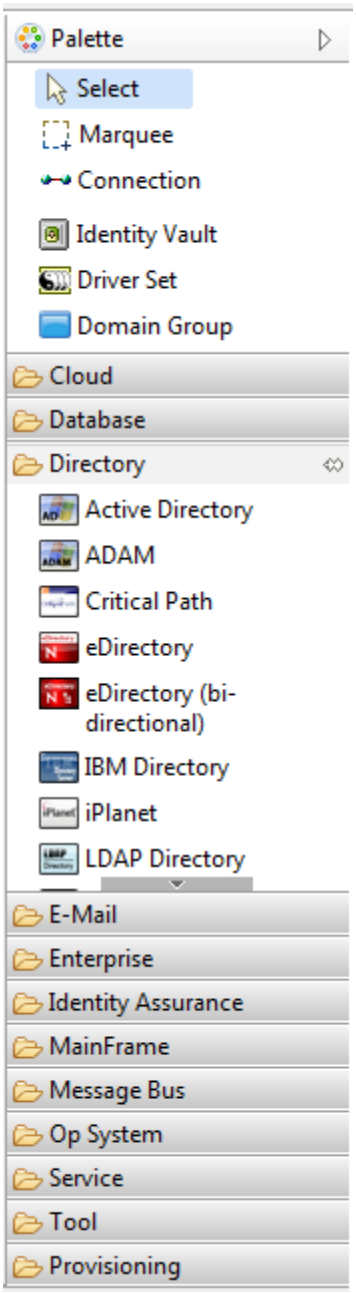
Template	Description
IdmUtil.ItemNumbering	Gets the current item numbering in context to the Designer source (such as "2.4.5.2."). This template helps centralize what should be counted in the numbering process because several places reuse this information.
IdmUtil.ItemText	Based on the XSI type, returns text for the type, followed by a colon and the name value (for example, Identity Vault: my vault 1)
IdmUtil.ItemType	Returns text representing the type of the current Item (such as Identity Vault, Domain, or Driver Set)
IdmUtil.PolicySetPropertyRow	Builds a property row with a list of the policies based on the next policy value. Parameters: ◆ policy - Root policy of the policy set, passed by attribute name (such as mappingPolicy). ◆ label - Label for the displayed value. ◆ emptyLabel - Text to show if the value is empty. The default value is (none defined).
IdmUtil.PolicySetLinks	Returns a list of policy set links, called recursively. Parameters: ◆ xsiRootPolicyHref - Root policy of the policy set.
IdmUtil.ConfigValuesTable	Shows a Config Value table for the given XML. Parameters: ◆ title - Title text. ◆ description - Description text. ◆ xml - XML value to use to create the table. ◆ border - Border used for the table. The default value is 0.5pt solid black. ◆ emptyLabel - Text to show if the value is empty. The default value is (none defined).
IdmUtil.FilterTable	Shows a Filter table for the given XML. Parameters: ◆ title - Title text. ◆ description - Description text. ◆ xml - XML value to use to create the table. ◆ emptyLabel - Text to show if the value is empty. The default value is (none defined).

Template	Description
IdmUtil.showSyncIcon	Show an OpenText Identity Manager sync icon based on input type and sub-type. Parameters: ◆ type - Type is pub, sub. ◆ sub-type - Sub-type is "", sync, ignore, notify, reset.
IdmUtil.ValueOfReferencedItem	Returns the value of the node given the XSI expression. When extracting the name of an item, you should use the Format.BasicLinkToReferencedItem method so that the text is created as a link inside the document. Parameters: ◆ xsiHref - XSI value of referenced node. ◆ suffix - Suffix to append before selecting (the default is the current node). The default value is "."
IdmUtil.ItemCustomIconFileName	Get the custom icon filename for the given GUID. Parameters: ◆ guid - The item's GUID
IdmUtil.ShowIManagerIcon	This method is for backwards compatibility. Use IdmUtil.ShowIcon instead.
IdmUtil.ShowIcon	Shows the icon for the current Item. This first checks the cusomIconURI for a referenced image, then builds to a generic path based on the type attribute (for Drivers and Applications). Parameters: ◆ image-width - The image width to use. The default value is 49px.

C Adding Applications and Drivers to the Palette

The following graphic illustrates Designer's palette. The Directory group is expanded to illustrate applications in that group.

Figure C-1 *Designer's Palette*



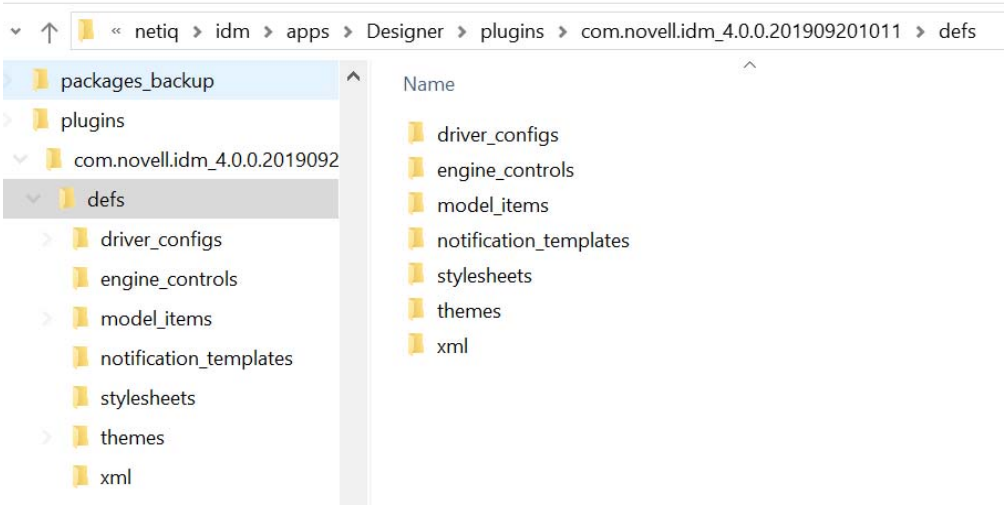
You can add a group, application, driver, or driver configuration to the palette. To do so, you must modify or create almost all the file types discussed in “Definition Folders and Files” on page 590. You must also exactly follow each step as explained in “Adding to the Palette” on page 599.

- “Definition Folders and Files” on page 590
- “Adding to the Palette” on page 599
- “Protecting Your Customized Files” on page 611

Definition Folders and Files

The palette definition is stored in the `com.novell.idm_<version and timestamp>’` folder.

Figure C-2 *The defs Folder*



The following sections provide information about subfolders and files:

- “Driver Configuration and Localization Files” on page 590
- “Palette Folders and Files” on page 590
- “The Notification Templates Folder” on page 598
- “The Themes Folder” on page 598

Driver Configuration and Localization Files

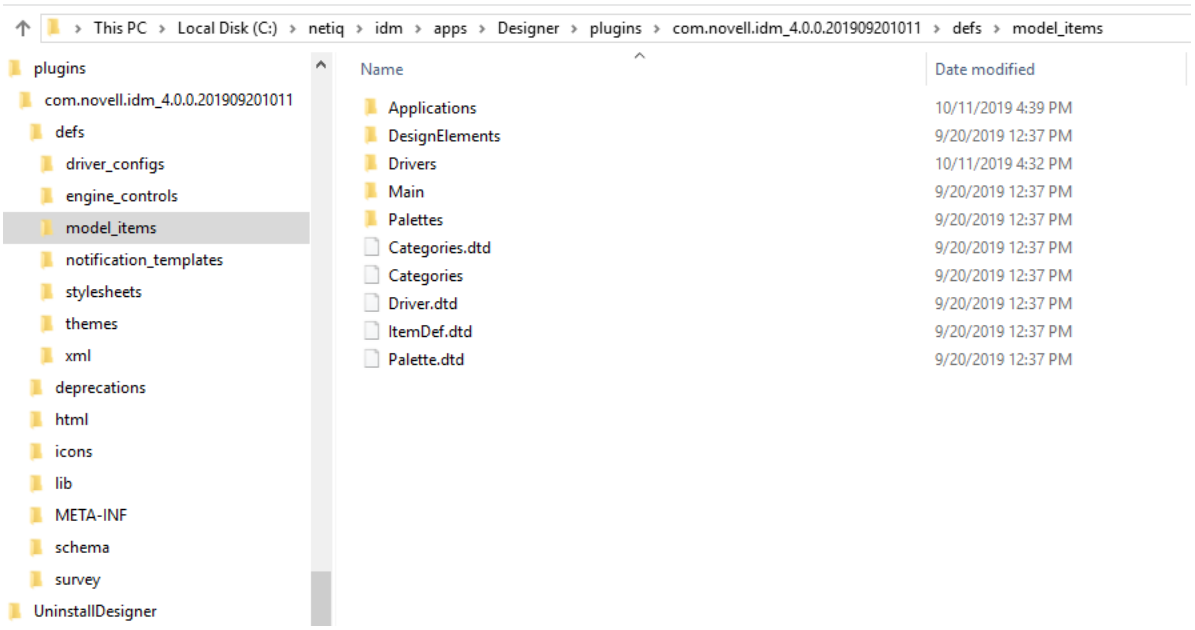
The `com.novell.idm_<version and timestamp>/defs/driver_configs` folder contains all the driver configuration files and their localization (`.xlf`) files. These files contain OpenText Identity Manager policies. You can import them by using iManager or Designer.

The `ids_transform` subdirectory should be left alone.

Palette Folders and Files

The `com.novell.idm_<version and timestamp>/defs/model_items` folder contains all the items that make up the palette:

Figure C-3 *The model_items Folder*



The following table lists the `.xml` and `.dtd` files found in this folder. The `.dtd` files contain the XML Document Type Definition for the different palette definition files.

Table C-1 *Files in the model_items Folder*

Filename	Description
`Categories.dtd`	Defines a category
`Categories.xml`	Contains all the categories that the palette can consume. Because adding or removing categories breaks existing code or has no impact at all, this file should be left alone.
`Driver.dtd`	Elements that make up a driver (for example, configuration files, primary and secondary applications, icons, and capabilities)
`ItemDef.dtd`	Defines applications and design elements
`Palette.dtd`	Defines the palette's name and its groups

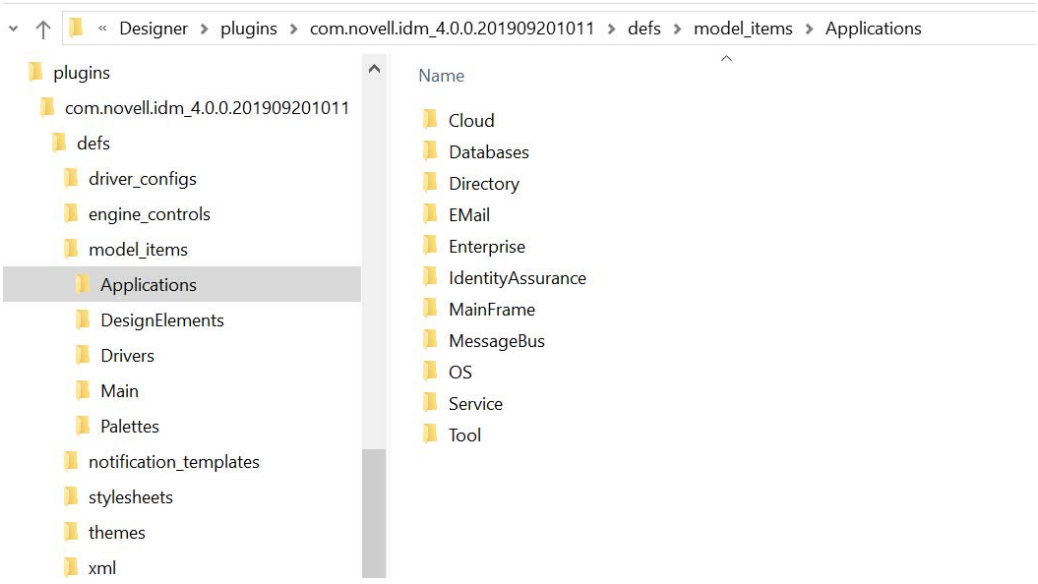
The `model_items` folder also contains several subfolders:

- “Definition Files for Applications” on page 591
- “Design Elements” on page 598
- “The Drivers Folder” on page 598

Definition Files for Applications

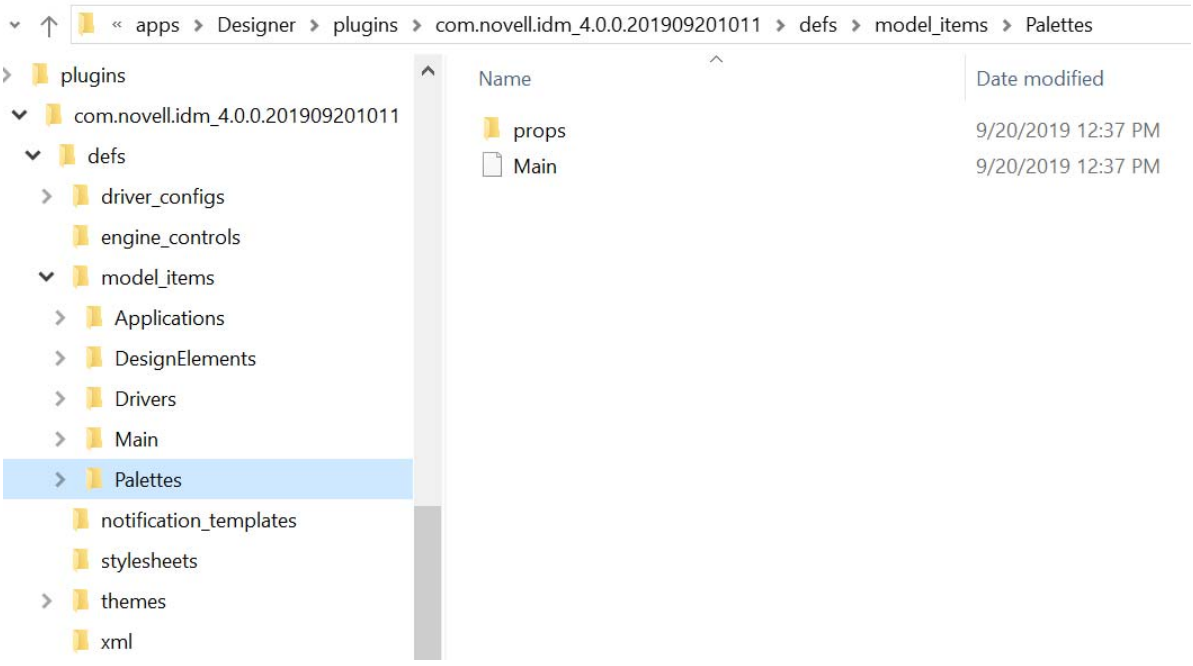
The `Applications` folder contains definition files for all applications that are available in the palette.

Figure C-4 *The `Applications` Folder*



The application definition files are grouped into folders that match the groups defined in the `Main.xml` palette definition file in the `Palettes` folder. The palette arranges applications in these same groups in Designer.

Figure C-5 *The Palettes Folder*



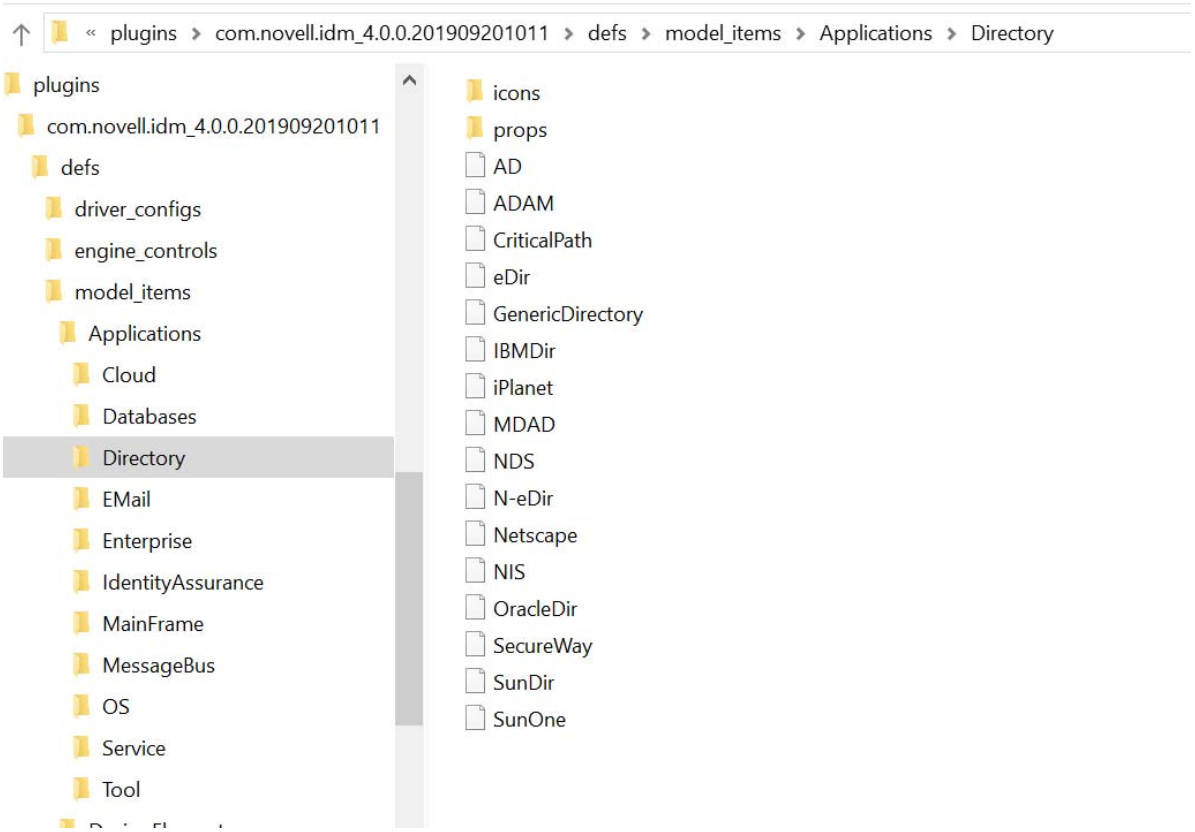
The `Applications/Directory` folder contains XML files, icons, and localization variables in properties files.

- "XML Files" on page 593
- "The Icons Folder" on page 594
- "Localization Files" on page 596

XML Files

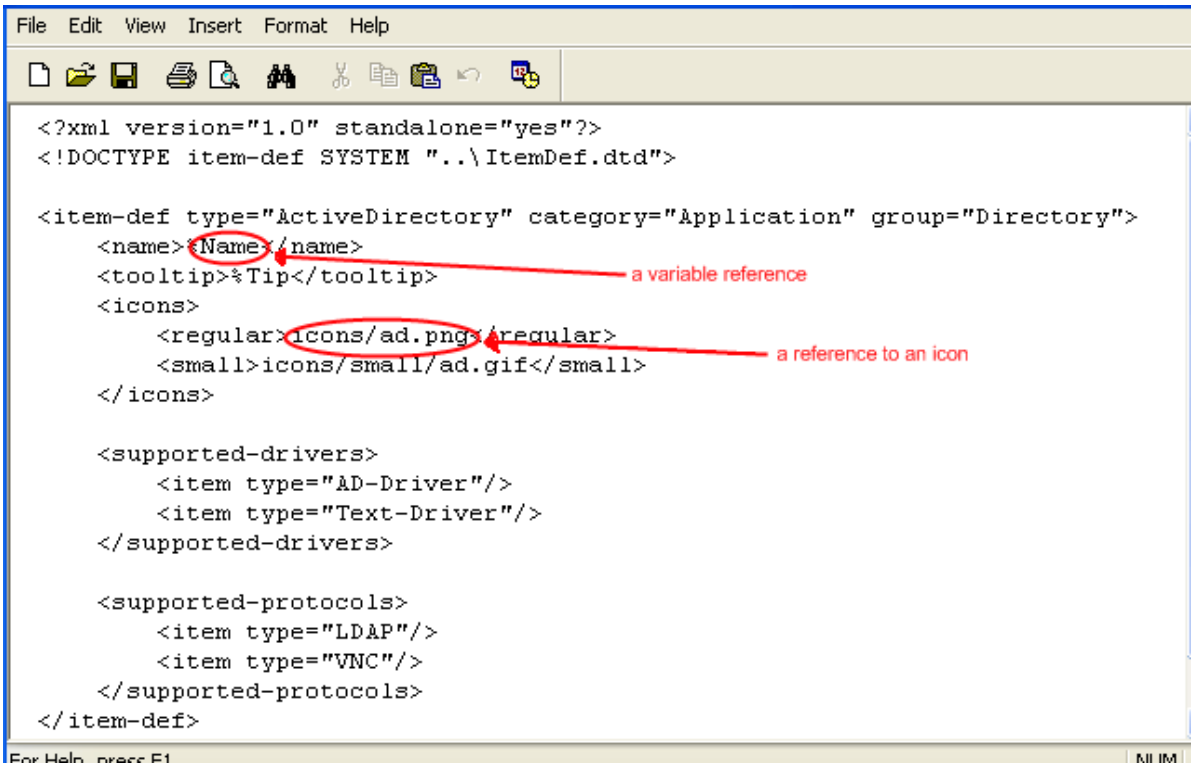
The `defs/model_items/Applications/Directory` folder contains `.xml` files. These files are the application definitions.

Figure C-6 *XML Files in the Directory Folder*



As the following figure illustrates, the definition files reference icons and localization variables:

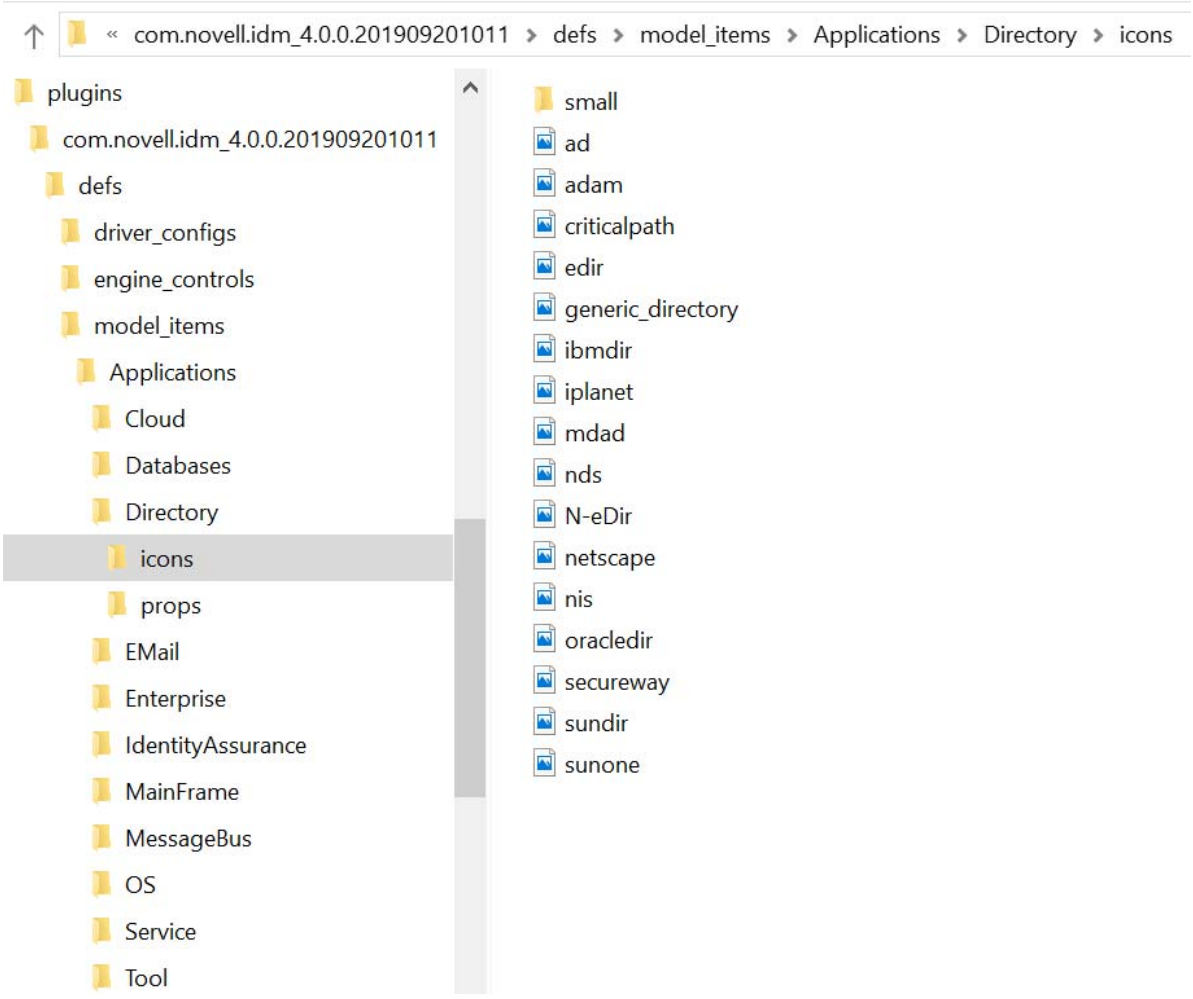
Figure C-7 *The AD.xml File*



The Icons Folder

The `defs/model_items/Applications/Directory/icons` folder contains icons in PNG format (`.png`).

Figure C-8 *The `icons` Folder*

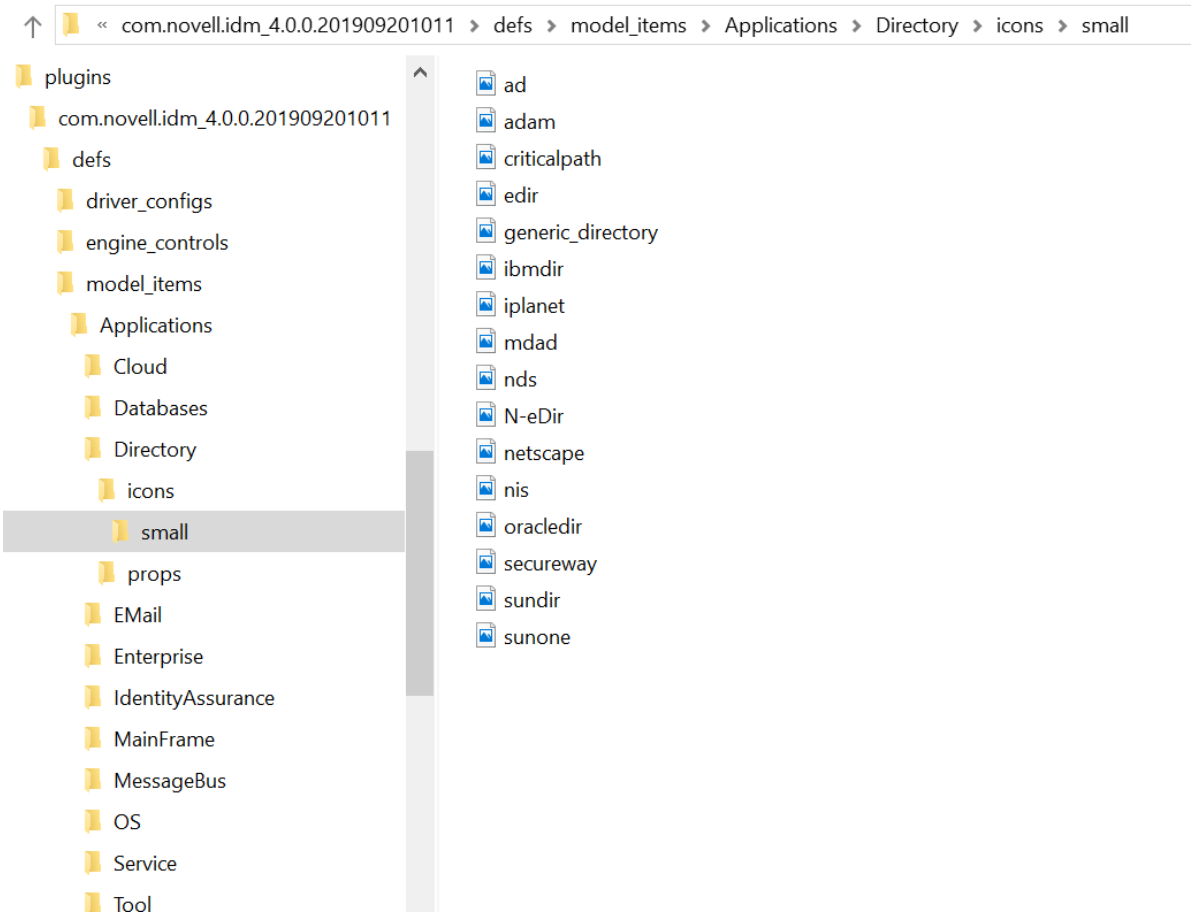


The Modeler (not the palette) uses these icons. The palette uses the small icons in the `small` subdirectory. The .png files are referenced from the application definition files.

The icons are 44x55 pixels in size and use transparency to display well in the Modeler.

The `small` folder contains smaller GIF versions of the icons in the parent directory.

Figure C-9 *The small folder*

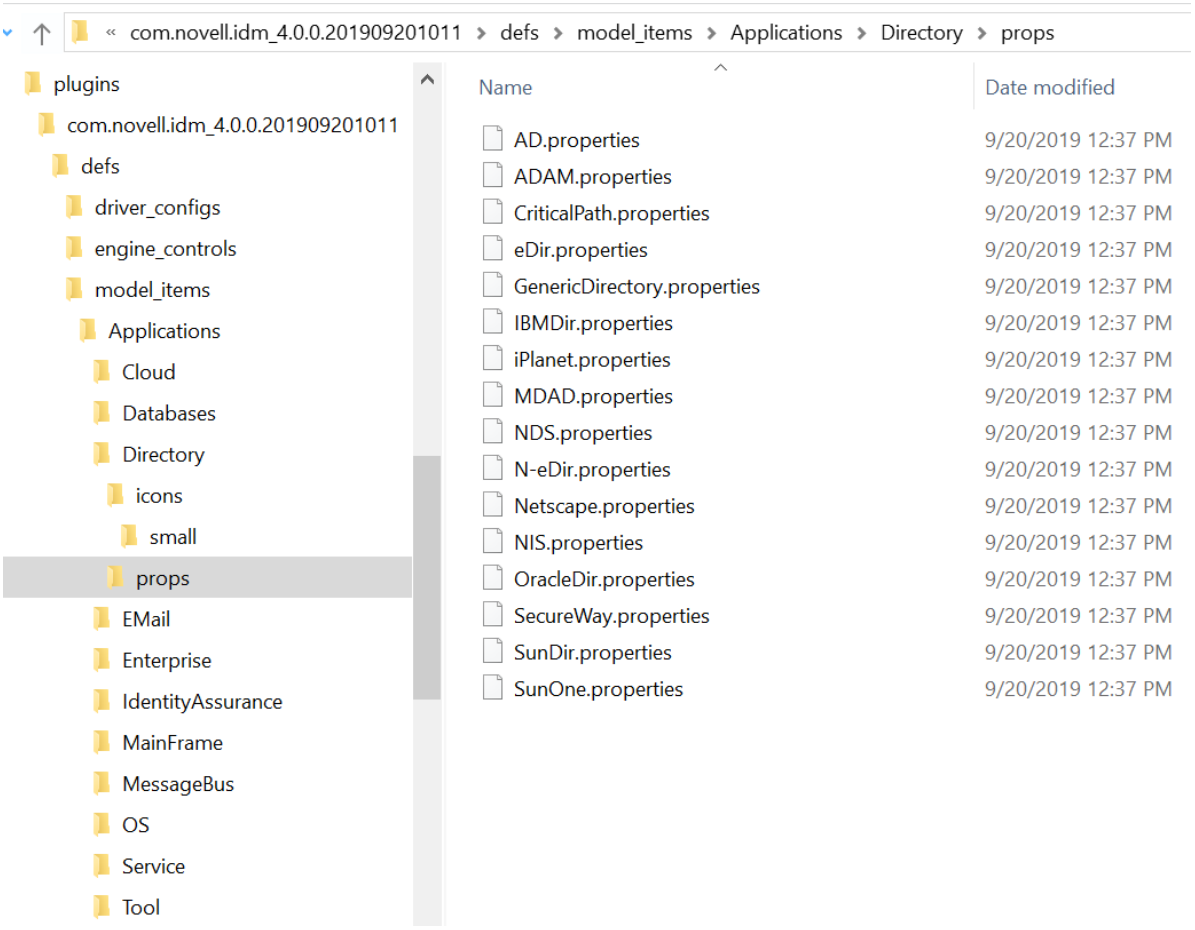


These icons are actually shown in the palette. The icons are 20x16 pixels in size and use transparency to display well in the palette.

Localization Files

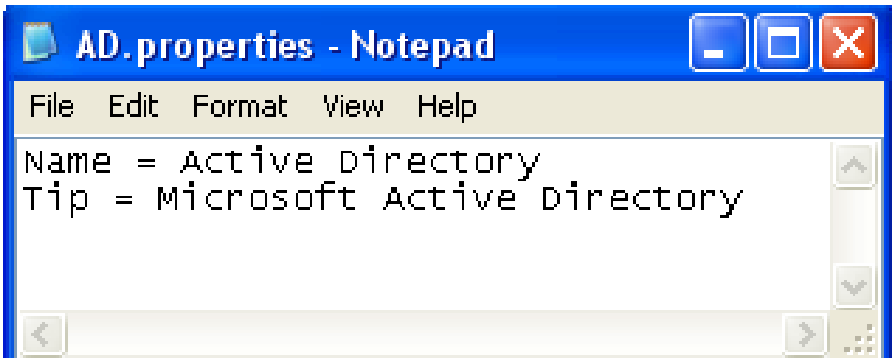
The `defs/model_items/Applications/Directory/props` folder contains localization variables that are defined in `.properties` files.

Figure C-10 *The `props` Folder*



The application definition files reference localized strings in the `.properties` files. In the following figure, **Name** can be referenced through %Name, illustrated in Figure C-7 on page 594.

Figure C-11 *A `.properties` File*



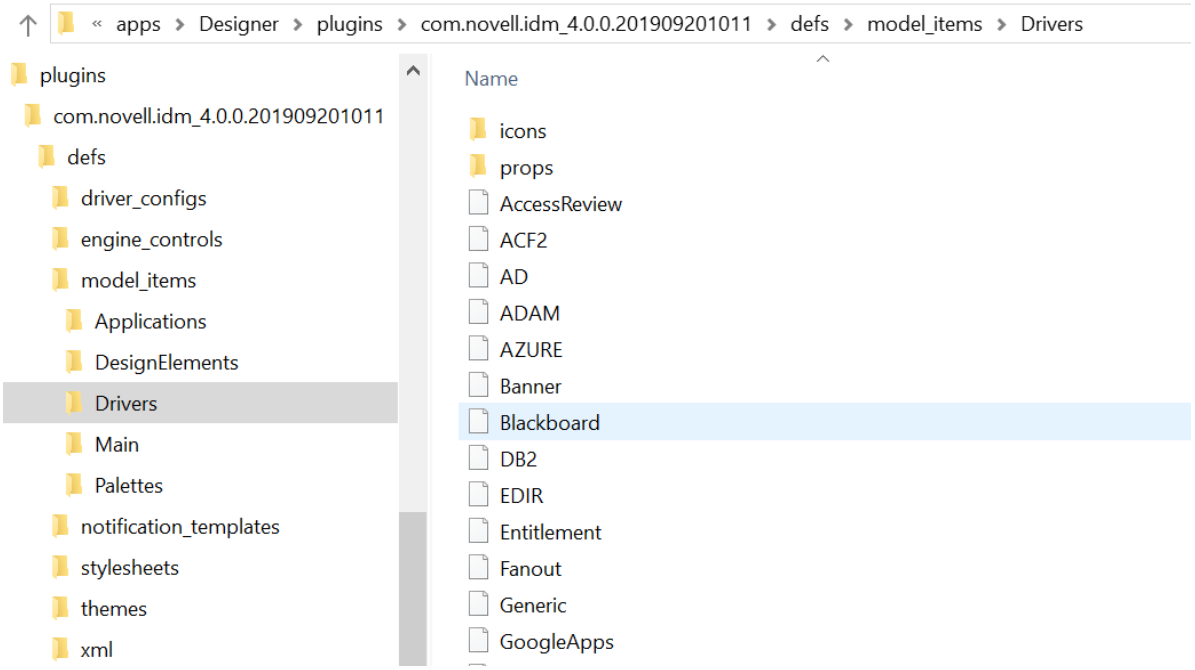
Design Elements

The `defs/model_items/DesignElements` folder mirrors the `Applications` folder, but contains design elements instead of applications. Design elements are like unknown applications to Designer. Design elements can be connected to and from anything, but Designer does not do anything with them. They have only a generic properties page, and no logic exists around them. They are basically just icons.

The Drivers Folder

The `defs/model_items/Drivers` folder contains the driver definition files (not the driver configuration files that contain OpenText Identity Manager policies and can be imported by using iManager or Designer).

Figure C-12 *The Drivers Folder*



The `icons` and `props` folders serve the same purpose as explained in "The Icons Folder" on page 594 and "Localization Files" on page 596.

The Notification Templates Folder

The `defs/notification_templates` folder contains the default e-mail notification templates that ship with Designer.

The Themes Folder

The `defs/themes` folder contains the Modeler theme definition files that ship with Designer.

Adding to the Palette

The need to extend the default palette usually arises when additional driver configuration files need to be hooked up to existing applications or to new applications or drivers.

Adding to the palette is a very delicate process and only successful if followed exactly step by step. Each step needs to be adapted to your situation.

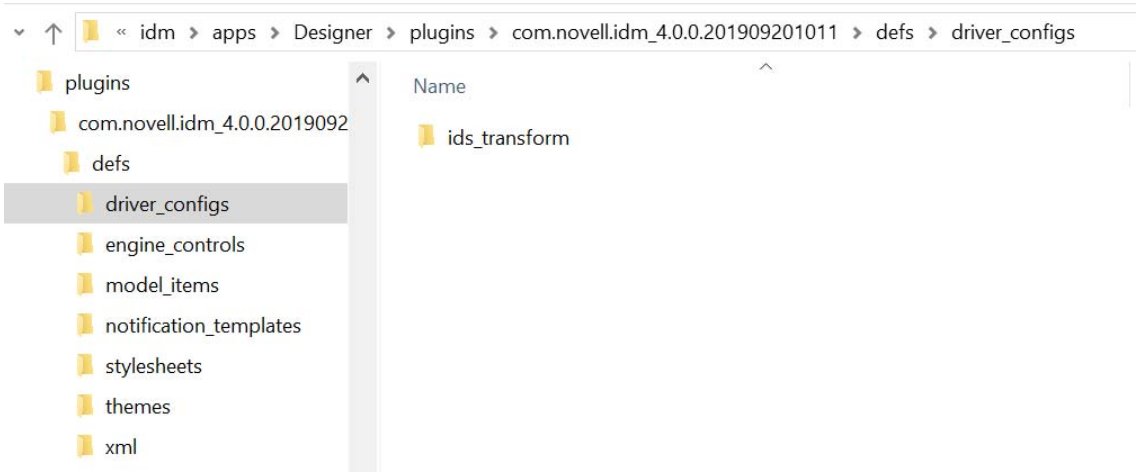
- "Copying Configuration Files" on page 599
- "Creating the Group" on page 600
- "Adding a Key_Value Pair" on page 600
- "Creating a Driver Definition" on page 602
- "Creating the Application" on page 606
- "Hooking Up the Custom Application" on page 609

Copying Configuration Files

1 Copy the new driver configuration file into the `driver_configs` folder so that the configuration file is accessible (but not yet hooked up) from Designer.

In this example, the new driver configuration file is `CustomDriver-IDM3_5_0-V1.xml`.

Figure C-13 *An Example New Driver Configuration File*



2 Copy into the `driver_configs` folder all corresponding `.xlf` files that belong to `CustomDriver-IDM3_5_0-V1.xml`.

3 Continue with "Creating the Group" on page 600 to connect the driver configuration file with the palette.

Creating the Group

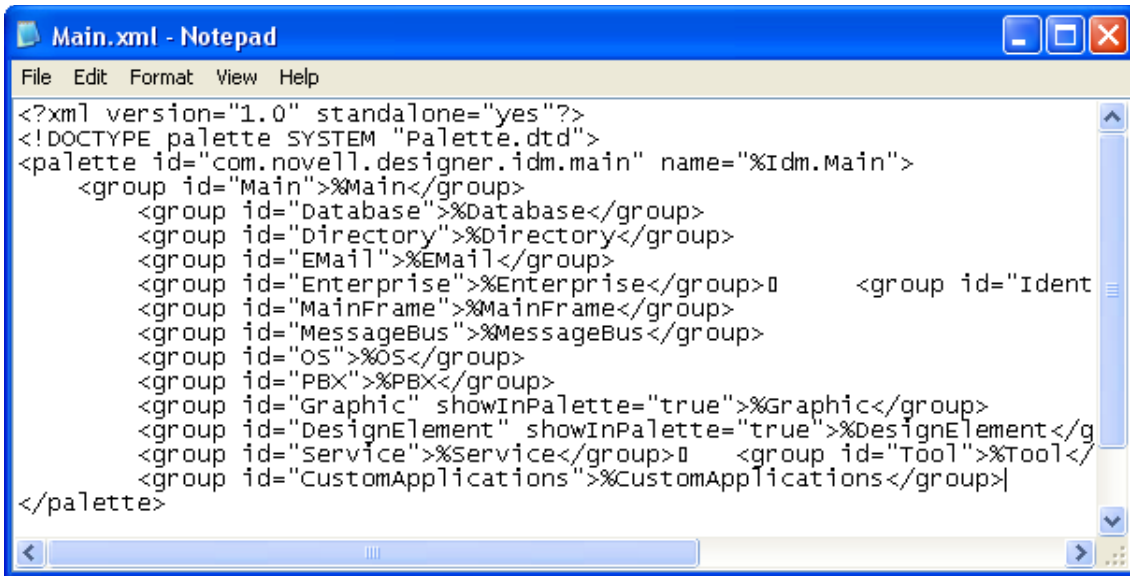
Before you place the new application Custom Application into the new Custom Applications group, you must first create the group.

1 Decide on the name of the new application that you want to create and the group that you want the new application to go into.

For this example, the following names are used:

- New application: Custom Application
- New group: Custom Applications

2 Add a group element to the `defs/model_items/Palettes/Main.xml` file.



Give the group element an ID attribute with an intuitive and unique value (for example, CustomApplications). Set the value of the element to `%CustomApplications` to make it localizable.

3 Save the file.

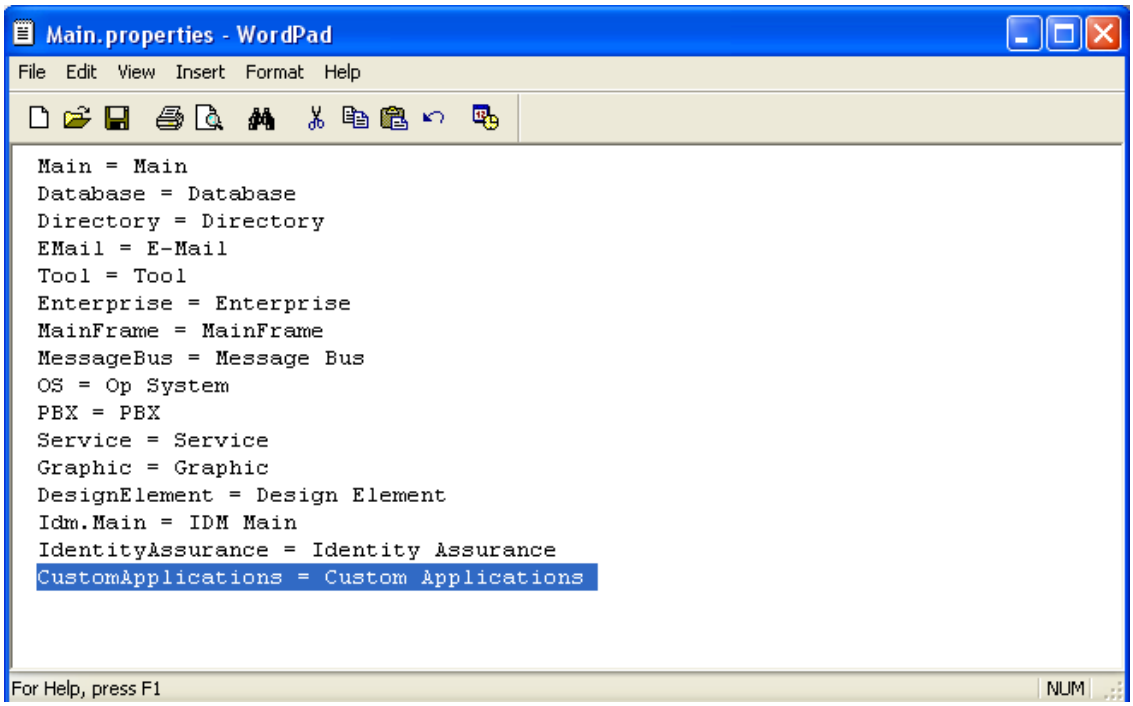
4 Continue to "Adding a Key_Value Pair" on page 600.

Adding a Key_Value Pair

1 Open `defs/model_items/Palettes/props/Main.properties`.

This is the properties file for the `Main.xml` file that you edited in Step 2 on page 600.

2 Add a key/value pair (for example, CustomApplications = Custom Applications).

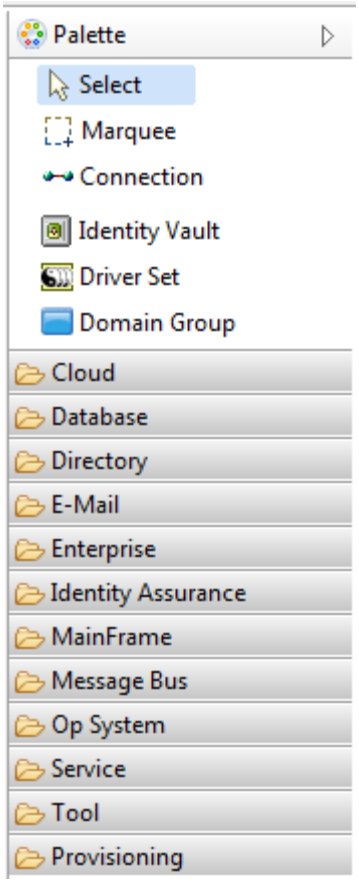


3 Save the file.

4 If you want to localize the group name into other languages, copy the properties file and rename it to `Main_`*`language_code`*`.properties`.

   For an example of supported languages and their codes, view the `.xlf` files in the `defs/driver_configs` folder.

5 View the new group as an empty group in the palette by starting the copy of Designer that you are manipulating.

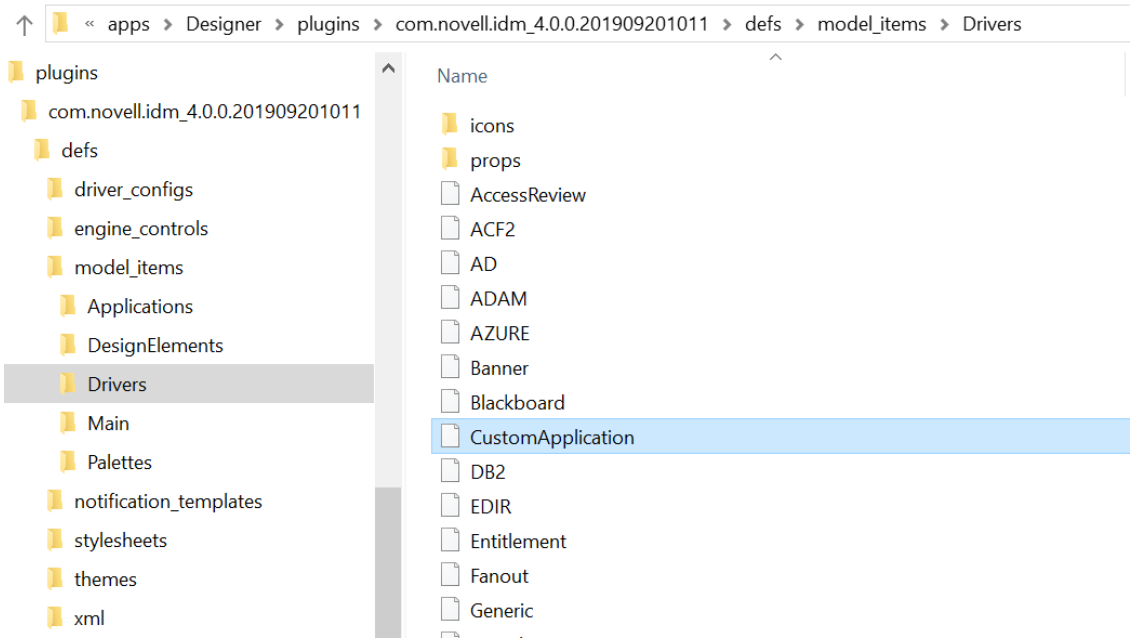


6 Continue with "Creating a Driver Definition" on page 602.

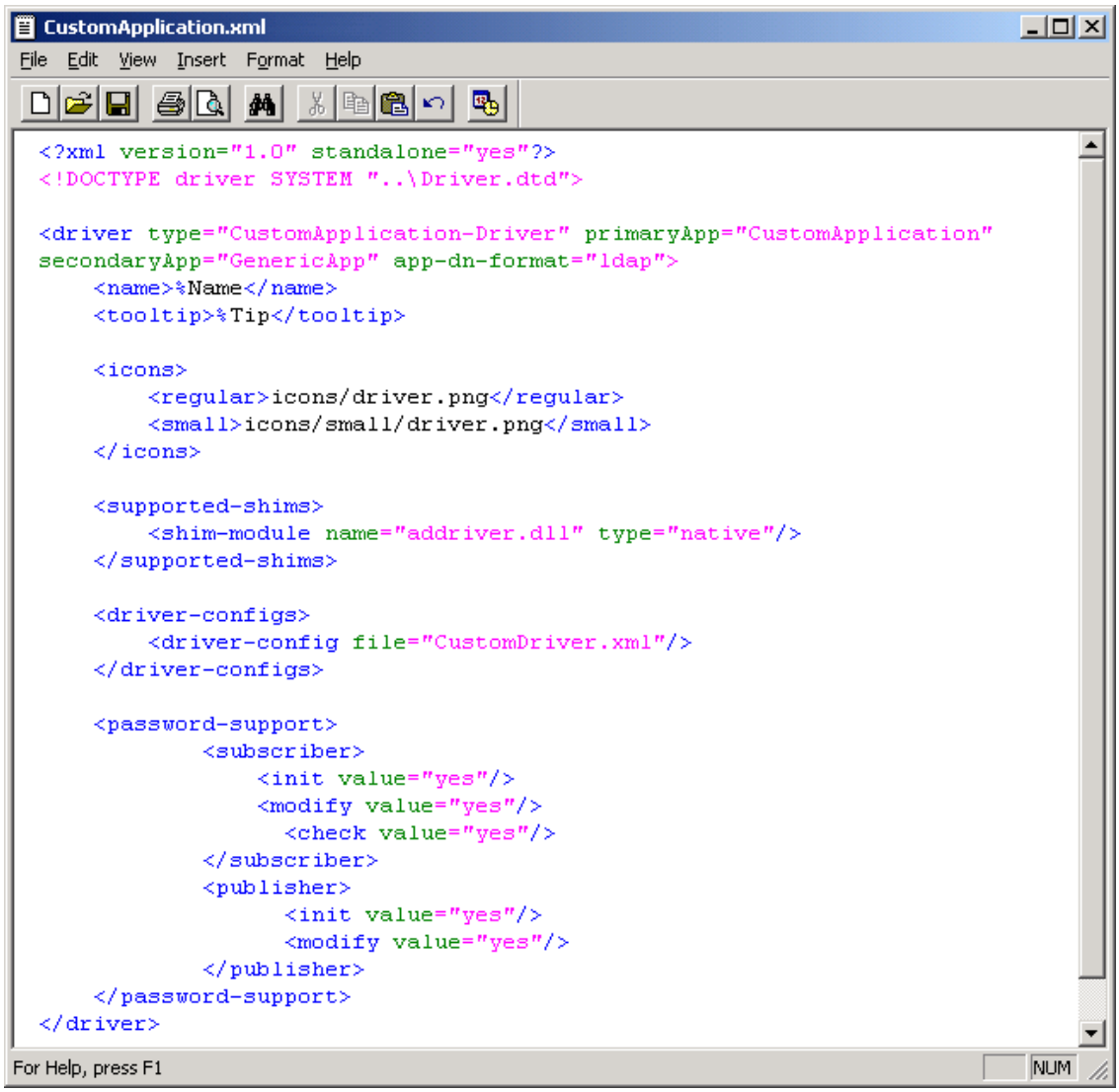
Creating a Driver Definition

1 Create a driver configuration file `CustomApplication.xml` in the `defs/model_items/Drivers` folder.

The new configuration file must follow the `Driver.dtd` specifications in the folder that you just created. The easiest way to do this is to copy an existing driver definition file, rename the file, then modify it.



2 Edit the configuration file.



2a Provide an intuitive and unique type (for example, CustomApplication-Driver).

2b Set the primaryApp value to CustomApplication.

2c Set secondaryApp value to GenericApp.

2d Specify the app-dn-format that your application supports.

2e Leave the icons as they are. They are not driver-specific.

2f Specify the shims that your application supports.

2g Specify the driver configuration file to use for this driver.

Specify only the filename, without the versioning information.For example, if your driver configuration file is named `CustomDriver-IDM3_5_0-v1.xml`, you refer to it as `Custom.xml`).

Because Designer hides or displays the user interface and features based on the version of the engine that you are working on, driver configuration filenames are important. You need to store the version information in the configuration filename, according to a well-defined format:

base name[*-type*]*-idm engine version-configuration file version* `.xml`

Examples:

- `ActiveDirectory-Mirror-IDM3_0_1-V9.xml`
- `ActiveDirectory-Flat-IDM3_5-V3.xml`
- `SAP-HR-IDM2_0_2-V2.xml`
- `SAP-User-IDM3_0_1-V1.xml`
- `SAP-User-IDM3_0_1-V2.xml`

In the example filenames, the IDM element identifies the engine version. The IDM elements to date are the following:

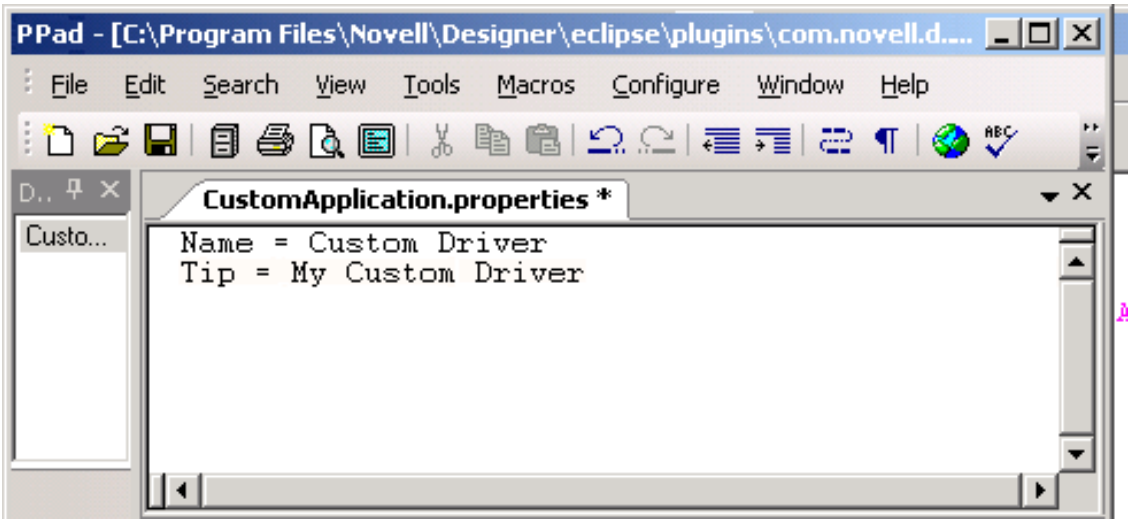
- IDM2_0
- IDM2_0_1
- IDM2_0_2
- IDM3_0
- IDM3_0_1
- IDM3_5
- IDM3_6
- IDM4_0

The V element in the example filenames specifies the version of this particular configuration file. It is a number that is incremented with each release of a new configuration file version. The following are examples:

- V1
- V9
- V11

No requirement exists for a more complex numbering schema.

3 Modify the `props/CustomApplication.properties` localization file.



You might need to create this file. If so, the quickest way is to copy, rename, and edit the file.

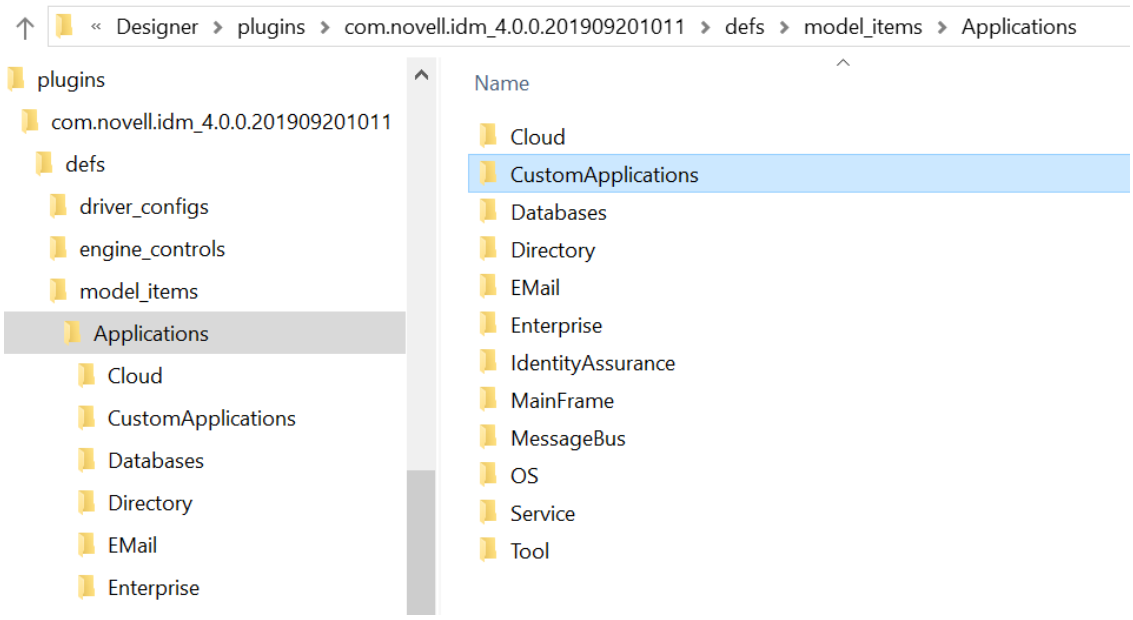
4 Continue with "Creating the Application" on page 606.

Creating the Application

The next step is to create the Custom Application application and place it in the new Custom Applications group.

1 Create a folder in the `defs/model_items/Applications` directory.

Name the folder the same name as the group ID. In this example, the name is CustomApplications, as specified in Step 2 on page 600.

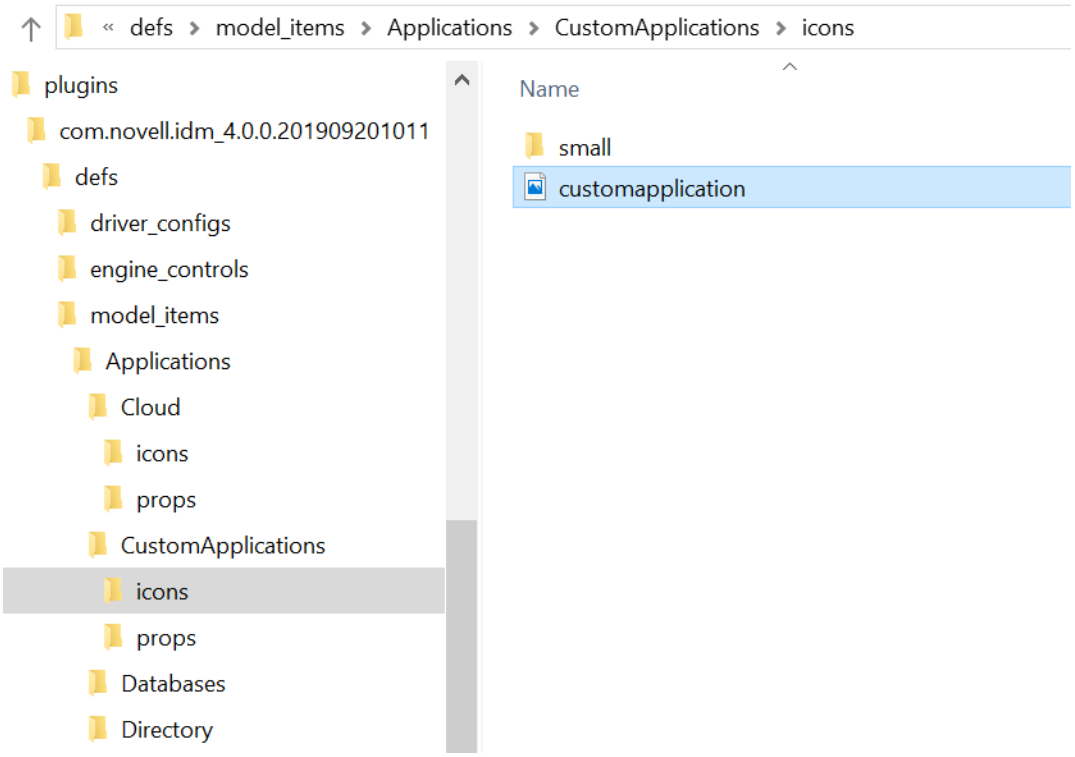


2 Create `icons`, `icons/small`, and `props` folders in the `CustomApplications` folder.

3 Create icons for the application.

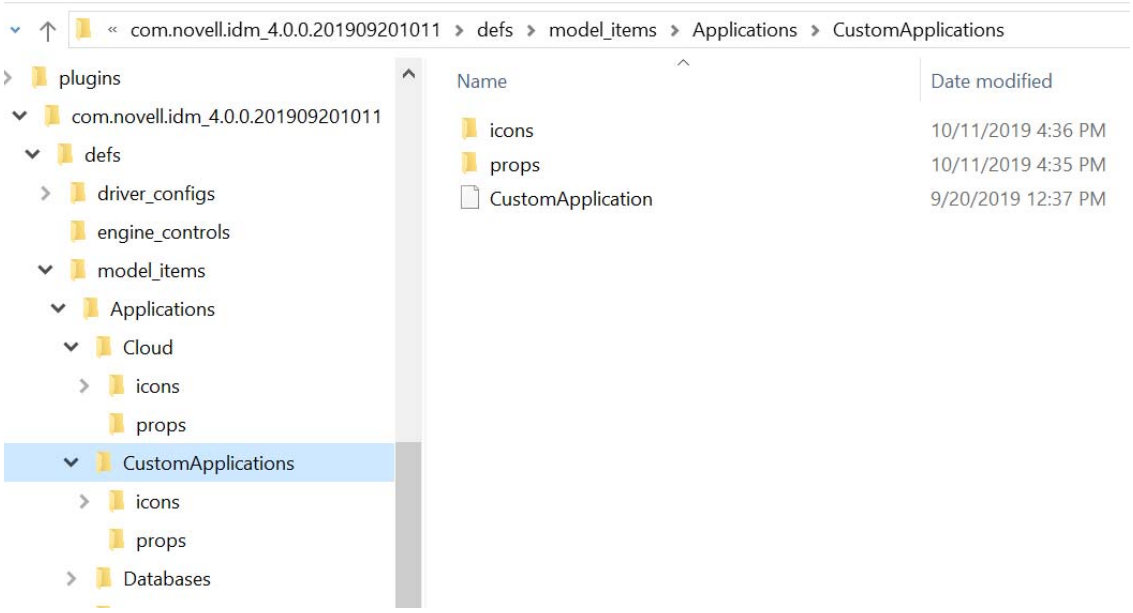
You can copy existing icons and modify them so the transparency is correct. In this example, modify the existing Generic Application icons.

3a Copy `defs/model_items/Applications/Tool/icons/generic_app.png` to `defs/model_items/Applications/CustomApplications/icons`. Rename the file as `customapplication.png`.



3b Copy `defs/model_items/Applications/Tool/icons/small/generic_app.gif` to `defs/model_items/Applications/CustomApplications/icons/small`. Rename the file as `customapplication.gif`.

4 Create an application definition file (`.xml`) in the `defs/model_items/Applications/CustomApplications` folder.

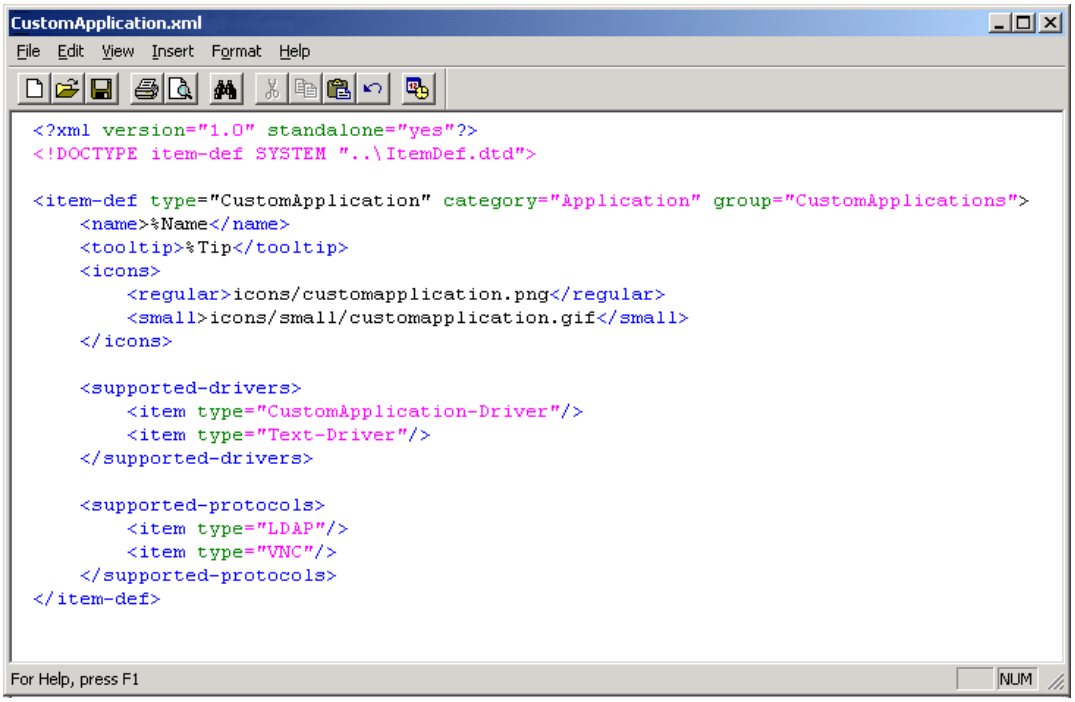


The definition file follows the `ItemDef.dtd` specifications. (See Table C-1 on page 591.)

The easiest way to create the file is to copy an existing application definition file (for example, `GenericApp.xml`), rename the file, then modify it.

The application definition file and the `.properties` file created in Step 5 need to have the same name as the type. In this example, the files are named `CustomApplication.xml` and `CustomApplication.properties`.

4a Make sure that the type attribute of the item-def element is set to an intuitive and unique name (for example, CustomApplication).



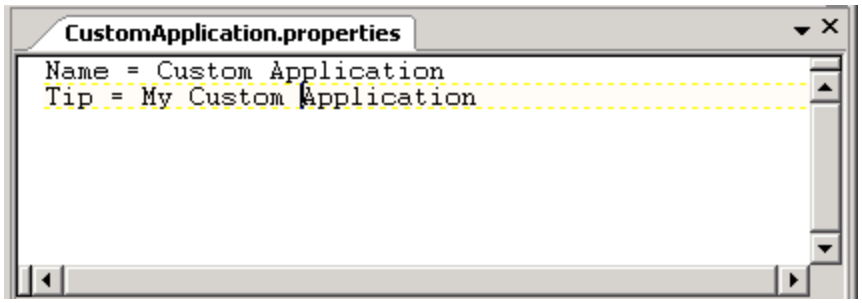
4b Leave the category attribute as **Application** and set the group attribute to the group ID, which is **CustomApplication**.

Reference the icons as you named them and do the same for the supported drivers. In this example, the Delimited Text Driver (Text-Driver) is added as an alternative to the Custom Application Driver (CustomApplication-Driver).

If the application can be connected to by using LDAP or VNC, leave these supported protocols in. Otherwise, remove them. Usually, every application runs on a host OS that supports either one or both of the protocols. Having these protocols registered enables certain functionality in Designer for that application.

5 Modify the `props/CustomApplication.properties` localization file in the same way you modified the group.

An easy way is to copy `defs/model_items/Applications/Tool/props/GenericApp.properties` to `defs/model_items/Applications/CustomApplications/props`. Rename the file to `CustomApplication.properties`, then modify and save the file.



6 Copy the `.gif` icon file into the `com.novell.designer.core/icons/iManager` directory.

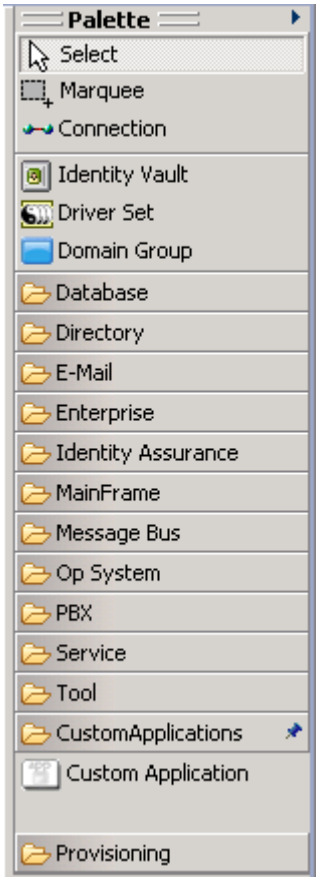
This icon is used in iManager after the driver is deployed into the Identity Vault.

7 Continue with "Hooking Up the Custom Application" on page 609.

Hooking Up the Custom Application

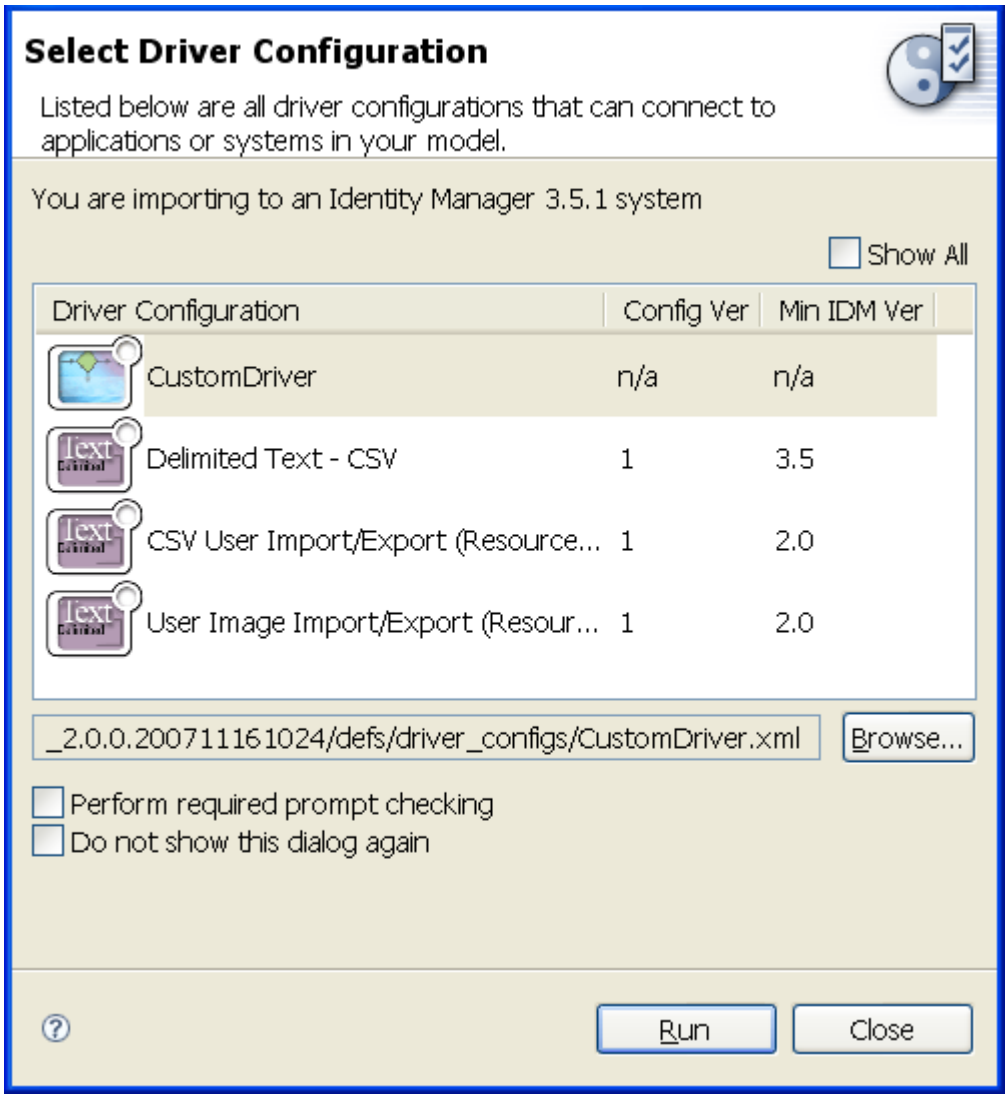
1 Run Designer.

The new Custom Application appears in the new Custom Applications group in the palette.



If you drag and drop the application to the Modeler workspace, the Driver Configuration Wizard prompts you to import the following:

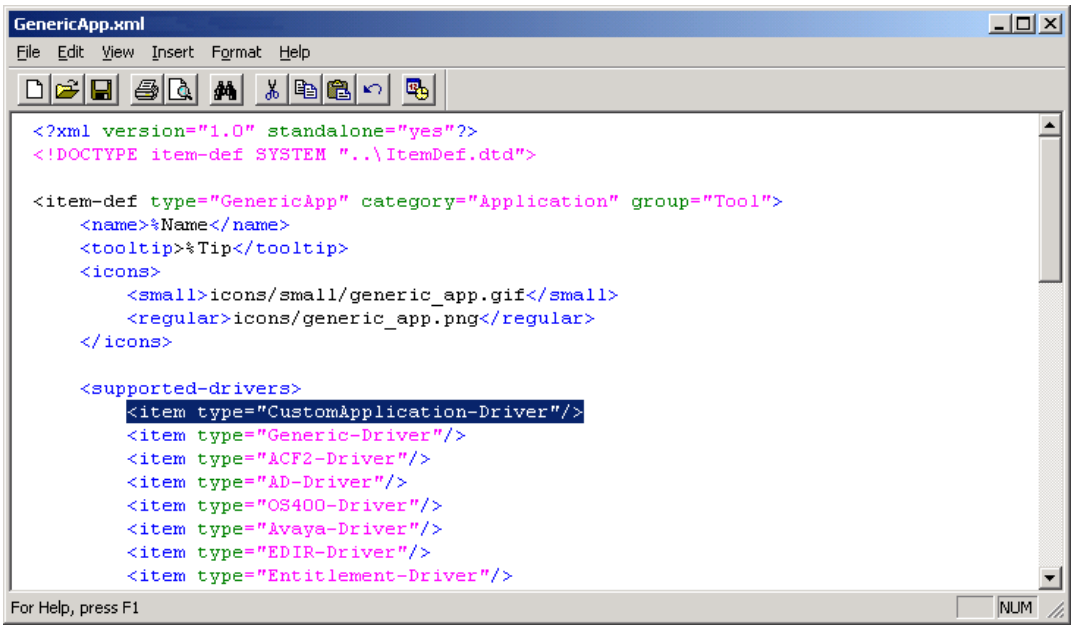
- The new Custom Driver configuration file
- All the Delimited Text driver configurations as specified in the application definition file



2 For full functionality in Designer, hook up your custom application to the Generic Application (GenericApp):

2a Open the application definition file `defs/model_items/Applications/Tool/GenericApp.xml`.

2b Add the new driver CustomApplication-Driver to the list of supported drivers.



If you now drag and drop a Generic Application from the Tools group, your new custom driver appears as a selectable option in the Driver Configuration Wizard.

Protecting Your Customized Files

The files that you created are customized files. If you upgrade Designer, you lose part of the customization in these files. Therefore, before upgrading, you need to save these customized files into a protected directory. After the upgrade, copy or re-create the files.

D Version Control with Subversion and Designer

This appendix is intended for people using Designer and Subversion. Designer includes complete documentation covering how to use version control. This appendix section gives more background on Subversion and indicates why you should make certain decisions. The Designer documentation tells you which protocols are supported. This appendix tells you why you should choose one over the others.

For more detailed information about using Designer with Subversion, see “Version Control” on page 445.

There are many books available on administering a Subversion server and working with Subversion. We recommend *Version Control With Subversion*. It is available without charge at O’Reilly Media (http://svnbook.red-bean.com). Many topics in the book are touched upon in this paper, and this paper references specific sections of the book.

- “Understanding Subversion” on page 613
- “Administering Your Subversion Server” on page 617
- “Taking Full Advantage of Version Control” on page 626
- “Glossary” on page 628

Understanding Subversion

Subversion is a version control system. Version control systems let you manage and create multiple revisions of your project and documents. They also allow you to share those revisions among a team of people.

- “How Revisions Work In Subversion” on page 613
- “Understanding Atomic Commits” on page 615
- “Where Subversion Stores the Project Data” on page 616
- “Moving an Existing Project” on page 616

How Revisions Work In Subversion

Revisions are a the heart of the Subversion functionality. A revision is a number that marks a specific set of changes made to a set of files. A single revision number can cover changes made to multiple files, but all of those files must be in the same repository.

Subversion uses a single revision number for the entire repository. This revision number is incremented every time any change is made to the Subversion server. For example, if you import a project at revision 100 and then create an Identity Vault and commit (revision 101), create a driver

and commit (revision 102), and create a policy and commit, you are at revision 103. If you have multiple projects in the same repository, every change made to any project increments the revision number for the whole server.

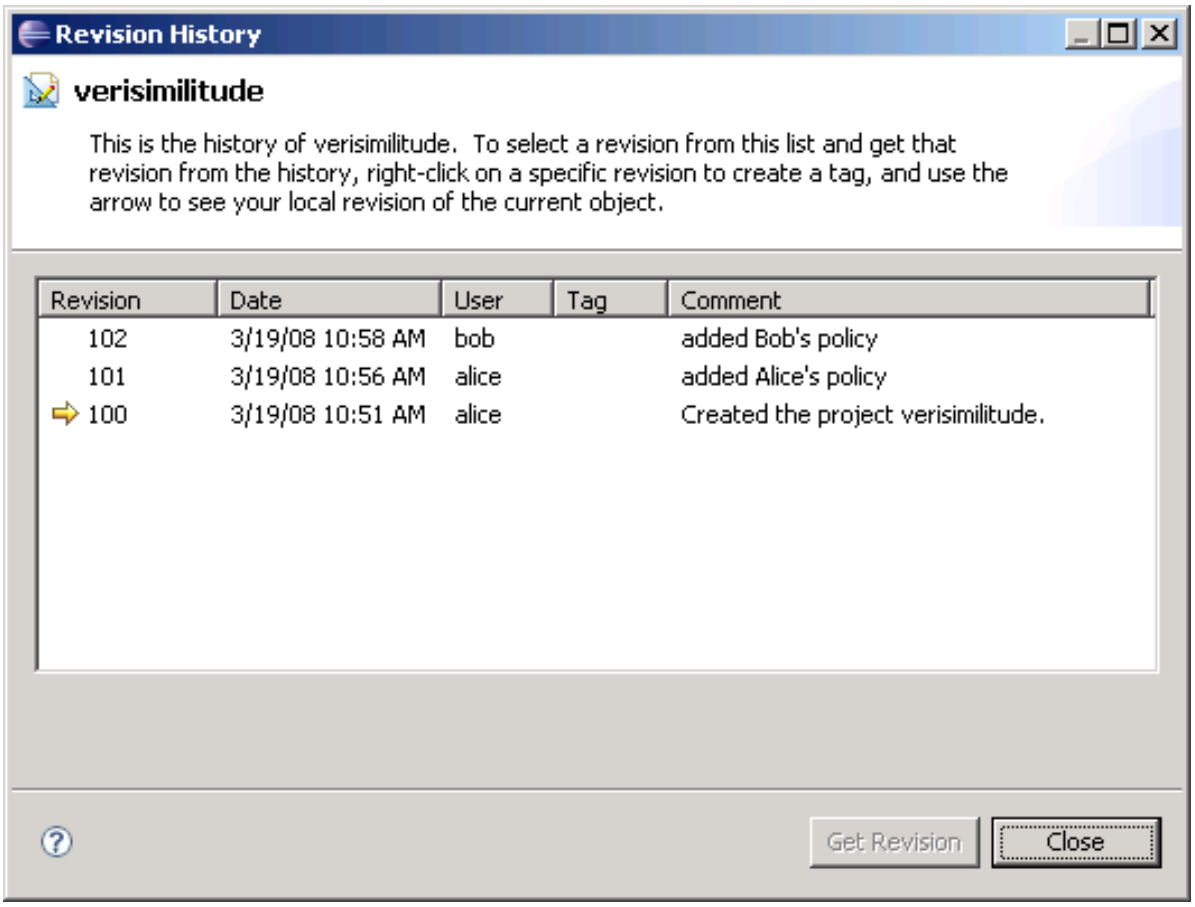
Although revision numbers are created for the entire server, different objects in your project can have different revision numbers. For example, suppose you start with revision 100 and create a policy and commit it; then create a mapping table resource and commit that version. The project will be at revision 100, the policy will be at revision 101, and the mapping table resource will be at revision 102. You can see the current revision of a specific object by using the Revision History or Properties page. The Revision History page indicates the specific object revision with a yellow arrow. In this example, the yellow arrow points to revision 100 for the project even though you see revision 101 and 102.

Subversion is meant to work in a team environment. In a team environment, there could be someone else editing the project at the same time as you. Let's look at an example:

- Alice imports a project at revision 100 to her local workspace.
- Bob imports the same project, also at revision 100, to his local workspace.
- Alice adds a new policy and commits, which creates revision 101.
- Bob adds a different new policy and commits, which creates revision 102

At this point Alice’s project is at revision 100, her policy is at revision 101, and the latest revision on the server is revision 102. If Alice wants to see Bob’s policy, she needs to update her project so she has revision 102.

Figure D-1 *Viewing Changes Through the Revision History*



Revisions are a useful way to track the versions of your project. Revisions can help you get projects back from the history and make sure that two users have the same version of a project loaded.

Understanding Atomic Commits

Atomic commits are a major feature of Subversion. The atomic commits treat the commit operation as a single event that either completely succeeds or fails gracefully. That means all of your changes are committed to the server or none of them are. For example, Alice and Bob are working together on a project. Alice makes changes to multiple policies and entitlements that are all interdependent. While Alice is in the process of committing this change, her network connection goes down. Before Alice can connect to the server again, Bob does an update. Subversion ensures that Bob does not get a partial update from Alice. Because Alice had a problem during her commit, Subversion makes sure that none of the files are changed on the server. Alice can then perform the commit after her network connection is restored.

Atomic commits are a very powerful tool and an excellent way to avoid broken projects. Atomic commits are always available within Designer and Subversion. You don't need to do anything special to enable them.

Where Subversion Stores the Project Data

When you commit a project to Subversion, the project is stored in the Subversion repository. The Subversion repository is an based on an internal database Subversion uses to store files. Subversion stores a separate file containing the specific changes made in each revision using the revision number as the filename.

These files are combined to maintain the concatenation of all of the changes made in your repository and the history of those changes. These files are iterative in nature and contain only the changes made for that specific revision. You can access these files in the `db/revs` directory of your Subversion repository.

Beyond that one requirement, there are no firm rules about setting up your projects. Here are some guidelines:

- It is a good practice to place a project in a directory of the same name. For example, a project called project1 would go into a folder such as `trunk/projects/project1`.
- Most repositories have a “sandbox” area. Users new to version control can experiment in this area without worrying about corrupting existing projects.
- It is a good practice to organize groups of projects. You can group projects by user, team, or company. The key is that having a large number of projects at the same level can be difficult to navigate.

Moving an Existing Project

Designer does not provide support for moving a committed project from one place on your Subversion server to another. However, you can do this with the Subversion command line:

- Make sure the whole team commits all of their local changes.
- Have all team members delete their local projects.
- Use the Subversion move command to move the project location.
- Have each team member import the project from the new location.

The Subversion move command is very simple. You just specify the current location of the project and the new location you want to move it to. For example, if your project is located at `trunk/project1` and you want to move it to `trunk/myprojects/project1`, use the following command:

```
 svn mv -m "<your comment for the move>" http://myserver/trunk/project1
http://myserver/trunk/myprojects/project1
```

Subversion moves the project to the new location and maintains all of the files and history.

Administering Your Subversion Server

Larger companies most likely have a Subversion server administrator. Smaller companies might require you to install the Subversion server yourself. You can also choose to install Subversion on your own machine for easy backups. Either way, it is a good idea to know how the server should be configured and administered.

- "Server Specifications" on page 617
- "Network Protocols" on page 618
- "Authentication Schemes" on page 621
- "Using Client Certificates" on page 623
- "Configuring Subversion with Apache HTTP" on page 624
- "Proxy Server Configuration" on page 624
- "Subversion Server Backup" on page 625

Server Specifications

The platform where you run Designer and the platform where you run the Subversion server are completely independent. Designer includes a Subversion client and is supported on any platform where Designer is supported.

Subversion provides official builds for the following platforms:

- Red Hat Linux
- Debian GNU/Linux
- FreeBSD
- OpenBSD
- NetBSD
- Solaris
- Mac OS X
- Windows NT, 2000, XP, and 2003
- HP-UX
- AIX
- IBM i5/OS (OS/400)

Subversion also works very well on SUSE Linux. Although OpenText strongly encourages you to run on SUSE Linux, the Subversion server works well on all of the platforms. The platform you choose might depend on the IT organization you are working with, existing infrastructure, or just personal preference.

Subversion is a lightweight product and doesn't require a very powerful machine. The specific requirements depend on many factors, such as the number of users, the number of projects, and the other software running on that system. There is a discussion thread with some specific recommendations you can find at the Apache Subversion Mailing Lists (http://subversion.apache.org/mailing-lists.html).

Network Protocols

Subversion supports direct file access, and the SVN, HTTP, HTTPS, and SVN+SSH network protocols. These protocols define how Designer communicates with the Subversion server. The server must be configured to support a set of specific protocols. You specify the protocol you are using in the first part of the URL you use to connect to your version control server.

The protocol that you are using is transparent while you use Designer. Everything works basically the same, no matter which protocol you use. However, the choice of protocol has significant impact on the network traffic, security, and speed of your interactions with Subversion. Choosing the protocol is an important decision.

- "Direct File Access" on page 618
- "SVN" on page 619
- "HTTP" on page 619
- "HTTPS" on page 619
- "SVN+SSH" on page 620
- "Protocol Comparison" on page 620

Direct File Access

Direct file access is not actually a network protocol. You can simply point Designer at a repository on your hard driver and access it directly. This is the easiest option to set up because it doesn't even require the Subversion server to be running. The version control import dialog box has an option to browse for your local repository location. This is a good option for single users, experimenting with version control, and giving demonstrations.

The main drawback of direct file access is that it doesn't support network access for multiple users. Direct file access is not a network protocol; your repository cannot be accessed by other people. As a result, it does not provide good support for authentication schemes. This makes direct file access a poor choice for team environments.

You specify this protocol by connecting to your server with a URL that looks like this:

```
C:\subversion\myrepository
```

or

```
/home/<my username>/subversion/myrepository
```

SVN

SVN is a Subversion-specific protocol. This is the protocol that is used when you run the Subversion server without the Apache HTTP Server. Just follow the Subversion server setup instructions in the Designer documentation and you are using this protocol. The SVN protocol supports networking and works well with small teams. It supports password file authentication as well as path-based authentication.

The SVN protocol does not support any type of encryption. This means that all information sent between Designer and the Subversion server is in clear text and could potentially be seen by a third party. Another concern with the SVN protocol is accessibility through firewalls. SVN is a specialized protocol and most firewalls need specific configuration to support it. Many firewall administrators are wary of changing their configuration.

You should check with all organizations involved before choosing this option. If you do need to configure a firewall to allow the SVN protocol, you must allow connections on TCP port 3690. In addition, the SVN protocol is not supported by most proxy servers.

The SVN protocol is a good choice for small teams where everyone works together in the same company. It is fast and easy to configure. You specify this protocol by connecting to your server with a URL that looks like this:

```
svn://mysubversionserver/myrepository
```

or

```
svn://localhost
```

HTTP

Subversion supports the use of HTTP by using a protocol called WebDAV. WevDAV allows Designer to access Subversion by using the same protocols that Web browsers use to access the Internet. The Subversion server also requires the Apache HTTP server to support the HTTP protocol. This requires a little more server configuration, but it isn't too difficult. Using the Apache HTTP server also allows many more authentication options.

The main advantage of HTTP is that it works with existing firewalls and proxy servers. This makes HTTP a good choice when working with multiple companies, or working inside corporate networks. HTTP does not support encryption between the Subversion server and Designer. If you need to protect your data, then you should choose a different protocol.

You specify this protocol by connecting to your server with a URL that looks like this:

```
http://subversion.mycompany.com/myrepository
```

HTTPS

HTTPS works very similarly to HTTP, with the addition of data encryption between the Subversion server and Designer. HTTPS uses the SSL (Secure Socket Layer) encryption protocol to make sure that third parties cannot read the communications between Designer and the Subversion server. HTTPS is

slightly slower, but in practice the difference is negligible. HTTPS is a good choice for corporate environments concerned about securing their data. HTTPS is the protocol OpenText developers use when working on the Designer source code.

The main drawback to HTTPS is that it can be difficult to configure. SSL requires a signing certificate that is granted by a certificate authority like Verisign.com. These certificates must be purchased, and applying for and installing them can be time-consuming. However, most server administrators are familiar with this process and should be able to guide you through it

You specify this protocol by connecting to your server with a URL that looks like this:

```
https://subversion.mycompany.com/myrepository
```

SVN+SSH

SSH (Secure Shell Protocol) is most popular on UNIX. Windows does not support SSH without additional software, and the configuration can be very difficult. SSH security is based on public key encryption using X.509 certificates. SSH is a good choice for UNIX environments looking for additional security. SSH requires a change to firewalls because it is not allowed on most corporate configurations. SSH uses TCP and UDP over port 22.

You specify this protocol by connecting to your server with and URL that looks like this:

```
svn+ssh://subversion.mycompany.com/myrepository
```

Protocol Comparison

Table D-1 *Protocol Comparison*

Protocol	Pros	Cons	Port
Direct File Access	Really easy to set up, great for single developers.	Doesn’t support team environments.	None
SVN	Easy setup and good network support.	Doesn’t support encryption, doesn’t support complex authentication, and has trouble with firewalls.	TCP 3690
HTTP	Good network support, works well with firewalls, and supports complex authentication.	Requires the Apache HTTP server and is not a good choice for running the Subversion server locally.	TCP 80
HTTPS	Good network support with good security options. A good choice for larger corporations.	Requires the Apache HTTP server, a certificate from a third party, and more complicated server configuration.	TCP 443
SVN+SSH	Good security in UNIX environments.	Doesn’t support Windows well and can be difficult to configure.	TCP/UDP 22

Authentication Schemes

In addition to deciding which protocols to use, it is important to look at authentication schemes. An authentication scheme defines the way users identify themselves to your Subversion server. This has significant impact on security as well as user management. Authentication schemes can be just a list of usernames and passwords in a flat file, or a multiple-server environment requiring special certificates for each client.

- "Specifying a Realm" on page 621
- "User Management" on page 622
- "Specifying Project-Level Access" on page 622

Specifying a Realm

Subversion makes use of realms in order to simplify user management. A realm is a string that identifies how your server authenticates its users. This string does not need to be unique to your server. Specifying the same realm in multiple servers indicates that the same username and password can be used in any server using that realm. The realm your server is using shows up when a user is prompted for authentication information in Designer.

Figure D-2 *Providing Authentication for the Realm*



By default, Subversion generates a unique ID for your realm, such as:

```
de409a8-8985-4647-ad92-44aef6788420
```

You can change the realm for your server in the `svnserver.conf` file located in your repository's `conf` directory. If you are using Subversion in conjunction with the Apache HTTP server, you need to use the Apache HTTP server configuration to specify your realm. More information about configuring this information can be found at the Apache Core Features page (http://httpd.apache.org/docs/2.2/mod/core.html#authname).

User Management

Whether you are just a single user or part of a large team, you need to manage the users who are allowed to access your Subversion server.

- "Flat Password File" on page 622
- "Apache HTTP Authentication" on page 622
- "Apache HTTP Authentication with Third-Party Modules" on page 622

Flat Password File

The easiest way to manage user access is with a password file. This file specifies a list of users and their passwords. The file looks like this:

```
[users] alice = alicepassword bob = bobpassword carol = carolpassword dave
= davepassword
```

This option is easy to configure and works well for small teams where security is not a major concern. However, in environments with larger teams, the management of this file quickly becomes unfeasible. In addition, this system is only as secure as the computer it is running on. If someone gains access to your Subversion server, they have access to this passwords file and every user's password.

Apache HTTP Authentication

If you configure Subversion to run with the Apache HTTP server, you can take advantage of the Apache HTTP server authentication. This mechanism also works with a flat file, but is much more flexible than the Subversion mechanism. This mechanism can manage users and groups, deny access by IP address, and much more. You can find information about this feature at Apache's Authentication, Authorization and Access Control for Apache HTTP Server page (http://httpd.apache.org/docs/2.0/howto/auth.html).

Apache HTTP Authentication with Third-Party Modules

Apache includes a large variety of third-party authentication modules. These modules support authentication to Windows NT domain controllers, UNIX password systems, OpenText eDirectory, and many more. OpenText uses a module to authenticate against an OpenText eDirectory server for its internal Subversion servers. As of this writing, there are 76 Apache HTTP modules dealing with authentication.

Creating a more complex authentication scheme might seem like a daunting task, but it can pay off in the long run. A good authentication mechanism can be mostly self-sustaining and gives users the opportunity to manage their own accounts. Combining advanced authentication with SSL or SSH provides ample security for a Subversion environment.

Specifying Project-Level Access

There are times when specifying access on a per-server basis is not sufficient. In those cases, you can use project-level access controls. There is support for this in Subversion as well as in the Apache HTTP server. When you configure this option by using the Subversion server, you can create an

authorization DB file. The following sample file grants Alice the rights to read and write everything, Bob the write to read everything, Carol the right to read and write **project 1** while only reading **project 2**, and Dave only the rights to read and write **project 2**.

```
[/] alice = rw bob = r

[/Project 1] carol = rw

[/Project 2] carol = r dave = rw
```

You must specify the location of this file by using the `authz-db` value in the `svnserve.conf` file in your Subversion repository conf directory. For more information about configuring this option with the Apache HTTP server, consult the documentation for the `mod_auth` and `mod_access` packages.

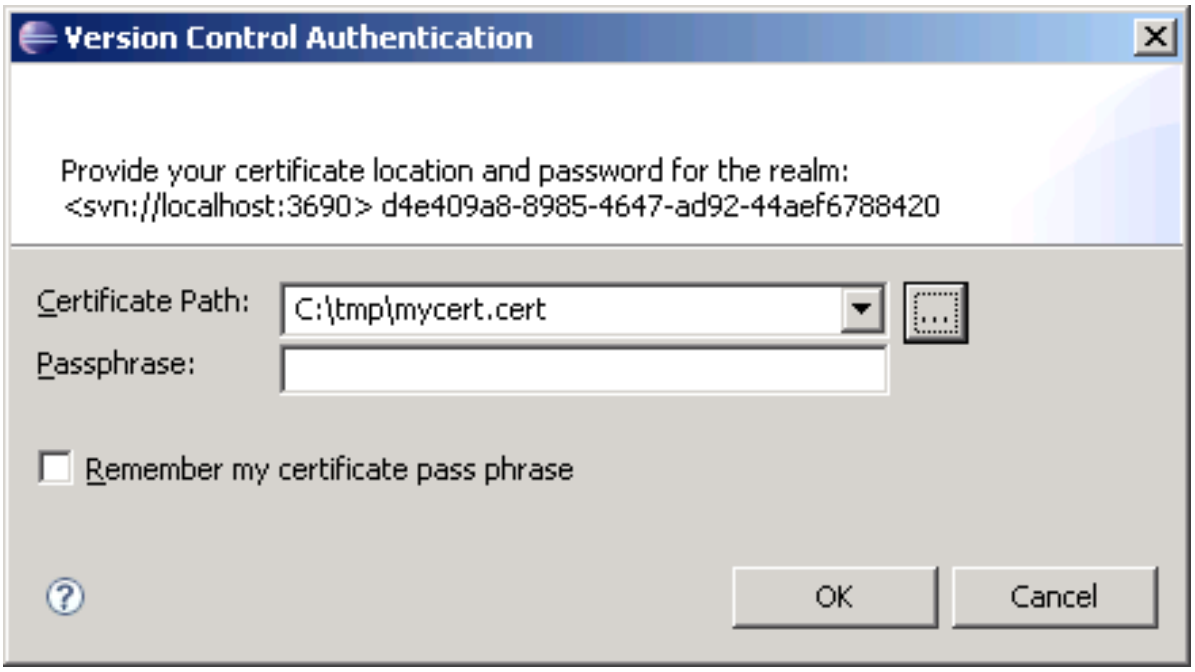
Using Client Certificates

Most security schemes in Subversion use a username and password to provide authentication. This is security based on something your know (your password). If you are especially concerned about security, you can use SSL client certificates. This is based on something you know (your password) and something you have (the certificate).

You can use client certificates with Designer and Subversion, but you must use the Apache HTTP Server. You will need to configure the Apache HTTP server to accept the client certificates. Apache can be configured to use client certificates by using the mod_ldap package. More information about that package can be found at the Apache Module mod_ldap page (http://httpd.apache.org/docs/2.2/mod/mod_ldap.html).

If your Subversion server is configured to use client certificates, you are prompted to provide a certificate in Designer. If you already have a Web browser configured to provide the client certificate, you can export the certificate for use with Designer. Tell your browser to export the client certificate and specify the PKCS#12 format. You can then browse and select this certificate when you are prompted by Designer.

Figure D-3 *Authenticating to Version Control*



Configuring Subversion with Apache HTTP

The Subversion server is a set of libraries. These libraries are accessible with the custom SVN protocol by using the `svnserve` program. They are also accessible with the HTTP and HTTPS protocols by using the `mod_dav_svn` module for Apache HTTP server. This is a module that knows how to use the Apache HTTP server to support Subversion functions by using the WebDAV protocol. You can find information about installing and configuring `mod_dav_svn` at mod_dav_svn Configuration Directives (http://svnbook.red-bean.com/en/1.1/re58.html).

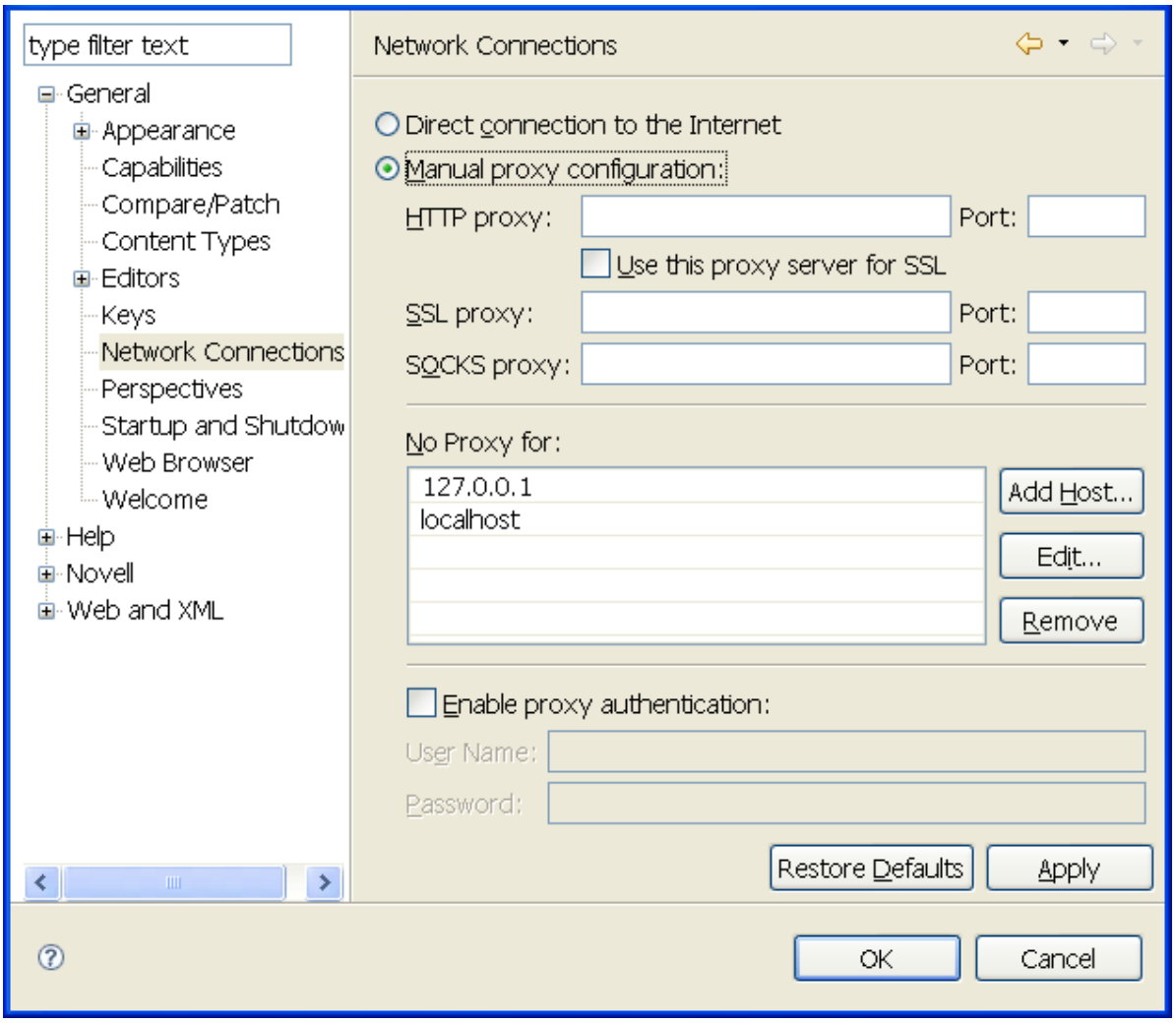
The standalone Subversion server is lightweight, easy to configure, and very stable. However, the Subversion server does not support HTTP, HTTPS, and advanced user authentication as well as other key features. The Subversion server is also not meant for large projects with many users. If you need any of the more advanced features, or if you need to support a large user base, you should use the Apache HTTP server. Both the Apache HTTP server and the Subversion server are free software.

Proxy Server Configuration

A proxy server is an application that takes requests and sends them on to other servers. Proxy servers are often used by companies to monitor and filter access to the Internet. Many large companies require all Internet access to be routed through the proxy server. If you are trying to access a Subversion server that is outside of such a network, you must configure the proxy settings in Designer.

In the main Designer menu, go to Window and then select **Preferences**. In the Preferences page, select **General > Network Connections**. This preference page allows you to configure the proxy server settings for Designer. Select the **Manual proxy configuration** option and supply the proxy settings specified by your network administrator.

Figure D-4 *Setting Proxy Server Settings*



Most proxy servers support only the HTTP and HTTPS protocols. Some proxy servers support the SVN+SSH protocol and almost none support the SVN protocol.

NOTE: If you use a proxy server, errors can occur occasionally when the proxy server fails to forward a packet. When errors occur, retry the operation. If you continue to have problems, verify that the proxy server is working correctly.

Subversion Server Backup

When you are using version control, the Subversion server acts as a backup mechanism for all your project data. It is vital that you back up the Subversion server frequently. If you do not back up the Subversion repository and your server has a hardware failure, you lose your project data. Daily backups are essential for active servers.

Subversion provides two tools to help create backups without interruptions of services: dump and hotcopy. The dump command takes your entire repository and sends the contents to standard out. You can also specify revisions to start and stop at. The hotcopy command creates a copy of your

Subversion repository, including the database and all other configuration information. You use the output from either of these commands to restore your Subversion repository during disaster recovery.

For more information about the dump and hotcopy commands, including examples, see svnadmin dump (http://svnbook.red-bean.com/en/1.1/re31.html) and svnadmin hotcopy (http://svnbook.red-bean.com/en/1.0/re33.html).

Taking Full Advantage of Version Control

Using version control to simply commit, update, and share projects can be very useful, but there is additional functionality that can be helpful in many of your projects.Version control can change the way you work. It can enable a truly team-oriented development methodology.

- "When to Commit and When to Update" on page 626
- "Comments" on page 626
- "Creating and Using Tags" on page 627
- "Subversion Keyword Substitution" on page 627

When to Commit and When to Update

Version control is a tool for sharing and backing up your project. You should take full advantage of it. That means committing often and updating frequently. You should learn to be comfortable with committing. The project doesn't need to be perfect, just make sure you won't impede your teammates.

You should also update frequently to get your teammate's changes. This ensures that you are working with an up-to-date project, and your changes can work with the changes your teammates are making. You also resolve conflicts in a better way. The earlier you can resolve a conflict, the easier it is be to resolve that conflict.

For example, if two individuals are editing the same policy and they work separately for a week, the two versions of the policy will be very different. This makes it very likely that there are conflicts and very likely that those conflicts are difficult to resolve. If those two users update frequently, they can avoid most of the conflicts and make them much easier to resolve.

Comments

Whenever you commit a change to the version control server, you are prompted for a comment. Comments are your chance to describe the change for yourself and for your teammates. Comments can explain why you did something and what you were thinking when you did it.

Good comments should take the form of sentences. They should describe what you did and why you did it. A well-written comment should give you a good idea of what has changed, but it does not need to describe every change in detail.

Good Comments

- Created a new project for work on the new Active Directory drivers for Unilateral Widgets Incorporated.
- Added a new AD driver to connect to the second directory and moved policy1 to a library so we can access it from the new driver.
- Changed the second rule in policy1 to avoid the potential for an infinite loop when handling more than three users.

Bad Comments

Comments should not be too brief:

- Added new policy
- New project
- Undid Joe's change

Comments should also not be too specific:

- Changed the condition of policy add password rule operation-data to be the following: <and> `<if-operation op="equal">add</if-operation> <if-password op="available"/> <if-xpath op="not-true">operation-data</if-xpath> </and>`

Creating and Using Tags

A tag is a readable name given to a specific revision. For example, you could tag revision 100 as *Release 1.0*. Tagging is most useful for identifying significant revisions. If you certify that you are ready to send a project to a customer, that is probably a good time to create a tag. You can then access that tag later if you need to roll back a change. The combination of tagging and the Get from History feature gives you a powerful tool to manage releases and deployments.

Subversion Keyword Substitution

You can use Subversion keyword substitution to give you more information about selected objects. For example, you can use the Description area to track the revision number, the date and time an object was last submitted to Subversion, and who submitted the last revision in the description of an object. The following example uses a policy.

1. Select a policy and bring it up in an editor.
2. In the Policy Description area, add the following keywords:
   - $Date$
   - Rev
   - $Author$
3. Save these changes to the policy.
4. Check the project into the version control server.
5. Update the project from the version control server.

6 Open the policy in an editor and you will see that Subversion has substituted the keywords to show the following:

- $Date: 2008-05-20 11:17:51 -0500 (Tue, 20 May 2008) $
- $Rev: 1513 $
- $Author: tpew $

You need to perform this procedure for each object for which you want to see this type of information. You can add keywords in any place where you can add text, but the object's Description area is most accessible. The keywords are updated as you or other team members make changes to the object.

Glossary

Appache HTTP Server. The server that is used in conjunction with the Subversion server to run the HTTP and HTTPS protocols. The Apache HTTP Server is free and open source. More information, including installation instructions, is available at the Apache HTTP Server Project Homepage (http://httpd.apache.org).

Authentication Scheme. Controls how users are authenticated with your Subversion server. Can be as simple as a flat file or can support many different authentication servers.

Commit. The process of taking your local changes and sending them to the Subversion server. Also called "checking in."

Import. You import a project from version control to get a project that has already been created and put it on your local machine. After the initial import, you can use the update process to download subsequent changes.

Repository. The place where the Subversion server holds your files. Repositories can support many projects, and many repositories can be supported on the same computer.

Revision. A number identifying a specific set of changes to files in your project and across the Subversion repository. Revisions are visible in the version control history and properties dialog boxes.

Server Realm. A string that identifies the authentication scheme to be used for that server. Servers with the same realms can use the same set of credentials to support a variation of single sign-on.

Update. Getting the latest changes from the server and applying them to the work area on your local machine.

E Additional Driver Configuration

Certain OpenText Identity Manager drivers require you to perform additional configuration in addition to the basic driver configuration. You need to perform this configuration for drivers such as Multi-Domain Active Directory driver and JDBC Fan-Out driver. To perform this, Designer provides new editors for configuring these drivers.

- "Configuring the Settings for the Multi-Domain Active Directory Driver" on page 629
- "Configuring the Database Connections for the JDBC Driver" on page 631

Configuring the Settings for the Multi-Domain Active Directory Driver

You can use the Multi-Domain Active Directory driver editor to accomplish the following tasks:

- Add forests and configure domain connections for the Multi-Domain Active Directory driver.
- Configure the driver with multiple domains within the same forest. The editor allows you to select the domains that you want to synchronize with OpenText Identity Manager.
- Configure a Primary Domain Controller (DC) and a list of alternate DCs for each domain.

  In case of a primary DC failure, the driver tries to establish connection with the alternate DCs.

Adding Forests to the Multi-Domain Active Directory Driver

You must first add the forests to configure the domain connections.

1. Open your project in Designer.
2. From the Palette, drag and drop the Multi-Domain Active Directory driver icon to the desired driver set in the Modeler.
3. In the Modeler, right-click the driver icon and select **Multi-Domain Active Directory Configuration**.

   The Multi-Domain Active Directory Configuration Editor displays.
4. Click the  icon to create a forest.
5. In the Add Forest pop up window, fill in the following fields:
   - **Forest Name:** Specify the forest name. Ensure that you specify a logical forest name that is accepted by the Identity Vault.
   - **Global Catalog Server:** Specify the global catalog server address. You can specify the port number along with the IP address. For example, `IP Address:port`. The default port for clear text is 3268 and for SSL is 3269.
   - **User:** Specify the username in LDAP format. For example, `CN=name,OU=employee,O=department`.
   - **Password:** Specify the global catalog server password.

- **Secure Connection:** Select this option to establish a secure connection with the global catalog server.

6 Click **OK**.

NOTE: This creates a new forest and adds the domains associated with the forest. By default, the root domain is added automatically. Designer displays the domains in the **Available Domains** list in the **Forest Configuration** tab.

---

7 Repeat step 4 through step 6 to create multiple forests for the Multi-Domain Active Directory driver.

Configuring the Domain Connections

After adding the forest, use the following steps to configure the domain connections for each forest.

1 In the **Forest Configuration** tab, select a desired domain from the **Available Domains** and move it to the **Selected Domains** list.

The selected domains also display in the Forest tree view.

2 Select the domain from the Forest tree view and proceed with the domain configuration.

3 In the **Domain Configuration** tab, fill in the following fields:

- **Domain:** Displays the selected domain name.
- **User:** Specify the username.
- **Wait Period:** Specify the interval that you want the driver to wait before re-establishing the connection with the next available domain controller during domain discovery failover. The default value is five minutes.
- **Domain Controllers:** Specify the domain controller configuration. The options are:
  - **Auto Discover:** The Multi-Domain Active Directory driver supports automatic DC discovery during driver runtime. Select this option to automatically discover the nearest DCs during driver startup.
  - **Configure Manually:** Select this option to configure the preferred and secondary domain controllers. To configure manually, select the desired domain controller from the **Available DCs** list and move it to the **Selected DCs** list.
- **Exchange-MDB:** Select the desired exchange mailbox database (MDB) that you want to provision to users in this specific domain from the **Available Exchange-MDB** list and move them to the **Selected Exchange-MDB** list. You can specify more than one mailbox database.
- **Trace File:** Specify the trace file. All the driver traces for this domain will be logged in this specified file. If you leave this field blank, the driver trace will be logged in the default trace file.
- **Trace Level:** Specify the trace level.
- **Trace File Size:** Specify the size of the trace file.

4 For the changes to take effect, click **Save** on the Designer toolbar.

IMPORTANT: After configuring the connection objects, deploy the connection object with the driver to the Identity Vault. After deploying it, link the connection objects to the **Subscriber options** in the Multi-Domain Active Directory driver configuration page.

Configuring the Database Connections for the JDBC Driver

You can create new database connections for the drivers that support fan-out configuration by using the Fan-Out Configuration Editor of Designer.

For example, you can configure multiple database connections for the Oracle Fan-Out driver. Each driver instance loaded by the Fan-Out Agent uses this database connection information to connect to the database and for tracing purpose.

To configure the database connections:

1. In the Fan-Out Configuration Editor page, click the  sign.
2. To add a new fan-out connection, specify the following information:
   - **Name:** Specify the name for the new connection.
   - **User:** Specify the user name with which the fanned out JDBC driver instance will authenticate with the database.
   - **Connection Password:** Specify the password required by the JDBC driver for authentication with this instance.
   - **Server:** Specify the server with which you want the fanned out instance to connect. For more information, see JDBC URL Syntaxes (https://netiq.com/documentation/identity-manager-4.10-drivers/jdbc/data/supported-third-party-jdbc-drivers.html#bvx364n) in the *OpenText Identity Manager Driver for JDBC Implementation Guide (https://www.netiq.com/documentation/identity-manager-4.10-drivers/jdbc/data/netiq-identity-manager-driver-for-jdbc.html)*.
   - **Trace Level:** Specify the trace level for the database connection. This defines the level for logging the trace messages.
   - **Trace File:** Specify the name of the trace file. This file includes the trace and debugging messages for the driver.
   - **Trace File Size:** Specify the trace file size. This defines the limit for the trace file.
3. For the changes to take effect, click **Save** on the Designer toolbar.
4. (Conditional) To create multiple database connection for the fan-out configuration, repeat Step 1 through Step 3.