opentext™

OpenText™ Identity Manager CE 24.4 (v4.10)

Using Designer to Create Policies

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 **13**

1 Overview **15**

Policies . . . 15

2 Managing Policies with the Policy Builder **17**

Accessing the Policy Builder . . . 17
- Model Outline View . . . 17
- Policy Flow View . . . 18
- Policy Set . . . 18

Using the Policy Builder . . . 19

Creating a Policy . . . 20
- Accessing the Policy Set . . . 21
- Using the Policy Set . . . 21
- Using the Add Policy Wizard . . . 22

Creating a Rule . . . 24
- Creating a New Rule . . . 25
- Using Predefined Rules . . . 27
- Including an Existing Rule . . . 28
- Importing a Policy From an XML File . . . 28

Creating an Argument . . . 29

Variable Selector . . . 31
- Dynamic Variable Expansion . . . 32
- Accessing the Variable Selector From the Conditions Tab . . . 32
- Accessing the Variable Selector From the Actions Tab . . . 33
- Accessing the Variable Selector From the Argument Builder . . . 34
- XPath Expressions . . . 35

Editing a Policy . . . 35
- Actions and Menu Items in the Policy Builder . . . 35
- Keyboard Support . . . 37
- Renaming a Policy . . . 37
- Saving Your Work . . . 38
- Policy Description . . . 38

Viewing the Policy in XML . . . 39

3 Using Additional Builders and Editors **41**

Action Builder . . . 41
- Creating an Action . . . 41
- Additional Options for the Action Builder . . . 42

Actions Builder . . . 42

Argument Builder . . . 43
- Launching the Argument Builder . . . 46
- Argument Builder Example . . . 47

Condition Builder . . . 48

Creating a Condition . . . 48
Additional Options for the Condition Builder . . . 49
Conditions Builder . . . 50
Match Attribute Builder . . . 51
Action Argument Component Builder . . . 53
Argument Value List Builder . . . 54
Named String Builder . . . 55
Condition Argument Component Builder . . . 56
Pattern Builder . . . 57
String Builder . . . 58
XPath Builder . . . 58
Mapping Table Editor . . . 58
Creating a Mapping Table Object . . . 59
Adding a Mapping Table Object to a Policy . . . 60
Editing a Mapping Table Object . . . 62
Importing Data from a CSV File . . . 63
Exporting Data to a CSV File . . . 63
Testing a Mapping Table Object . . . 63
Global Configuration Value Definition Editor . . . 64
Namespace Editor . . . 65
Accessing Java Classes Using Namespaces . . . 66
Local Variable Selector . . . 67

4 Using the XPath Builder 69

5 Defining Schema Map Policies 75

Using the Schema Map Editor . . . 76
Accessing the Schema Map Editor . . . 76
Navigating the Schema Map Editor . . . 77
Understanding the Schema Map Editor Toolbar . . . 78
Editing a Schema Map Policy . . . 79
Adding or Deleting Classes and Attributes . . . 80
Refreshing the Application Schema . . . 84
Editing Items . . . 84
Sorting Schema Map Entries . . . 85
Managing the Schema . . . 85
Testing Schema Map Policies . . . 86
Exporting and Importing with the Schema Map Editor . . . 86
Exporting a Schema Map Policy . . . 86
Importing a Schema Map Policy . . . 86
Accessing the Schema Map Policy in XML . . . 86
Additional Schema Map Policy Options . . . 87
Outline View Additional Options . . . 87
Policy Flow View Additional Options . . . 88
Policy Set View Additional Options . . . 89

6 Controlling the Flow of Objects with the Filter 91

Using the Filter Editor . . . 92
Accessing the Filter Editor . . . 92
Navigating the Filter Editor . . . 95

Understanding the Filter Editor Toolbar 95
Editing the Filter 96
Removing or Adding Classes and Attributes 96
Modifying Multiple Attributes 97
Copying an Existing Filter 97
Setting Default Values for Attributes 97
Changing the Filter Settings 98
Testing the Filter 102
Exporting and Importing Filter Files 102
Exporting a Filter File 102
Importing a Filter File 102
Adding Comments to Classes and Attributes 103
Viewing the Filter in XML 103
Deploying the Filter 103
Additional Filter Options 103
Outline View Additional Options 103
Policy Flow View Additional Options 104
Policy Set View Additional Options 105

7 Using Predefined Rules 107

Accessing the Predefined Rules 107
Command Transformation - Create Departmental Container - Part 1 and Part 2 108
Creating a Policy 108
Importing the Predefined Rule 108
How the Rule Works 109
Command Transformation - Publisher Delete to Disable 110
Creating a Policy 110
Importing the Predefined Rule 111
How the Rule Works 112
Creation - Require Attributes 112
Creating a Policy 112
Importing the Predefined Rule 113
How the Rule Works 113
Creation - Publisher - Use Template 114
Creating a Policy 114
Importing the Predefined Rule 114
How the Rule Works 115
Creation - Set Default Attribute Value 115
Creating a Policy 116
Importing the Predefined Rule 116
How the Rule Works 117
Creation - Set Default Password 117
Creating a Policy 117
Importing the Predefined Rule 118
How the Rule Works 118
Event Transformation - Scope Filtering - Include Subtrees 119
Creating a Policy 119
Importing the Predefined Rule 119
How the Rule Works 120
Event Transformation - Scope Filtering - Exclude Subtrees 120
Creating a Policy 120
Importing the Predefined Rule 121

How the Rule Works 122
Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn 122
Creating a Policy. 122
Importing the Predefined Rule 123
How the Rule Works 123
Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn 124
Creating a Policy. 124
Importing the Predefined Rule 125
How the Rule Works 125
Matching - Publisher Mirrored. 125
Creating a Policy. 126
Importing the Predefined Rule 126
How the Rule Works 127
Matching - Subscriber Mirrored - LDAP Format 127
Creating a Policy. 127
Importing the Predefined Rule 128
How the Rule Works 129
Matching - By Attribute Value 129
Creating a Policy. 129
Importing the Predefined Rule 130
How the Rule Works 131
Placement - Publisher Mirrored 131
Creating a Policy. 131
Importing the Predefined Rule 132
How the Rule Works 132
Placement - Subscriber Mirrored - LDAP Format 133
Creating a Policy. 133
Importing the Predefined Rule 133
How the Rule Works 134
Placement - Publisher Flat 134
Creating a Policy. 135
Importing the Predefined Rule 135
How the Rule Works 136
Placement - Subscriber Flat - LDAP Format 136
Creating a Policy. 137
Importing the Predefined Rule 137
How the Rule Works 138
Placement - Publisher By Dept. 138
Creating a Policy. 139
Importing the Predefined Rule 139
How the Rule Works 140
Placement - Subscriber By Dept - LDAP Format 141
Creating a Policy. 141
Importing the Predefined Rule 141
How the Rule Works 142

8 Testing Policies with the Policy Simulator 143

Accessing the Policy Simulator. 143
Outline View. 144
Policy Flow View 144

Editors . . . 144
Creating an XDS Input Document . . . 144
Source . . . 146
Import an XDS Document . . . 146
Use an Identity Vault Object As a Template . . . 146
Clear All Parameters . . . 147
Configuration Options . . . 147
Save the Input Document . . . 147
Simulation Point . . . 147
Operation . . . 148
Parameter and Value . . . 148
Attributes . . . 149
Using the Operation Data Editor . . . 152
Using the Hex Editor . . . 152
Accessing the Hex Editor . . . 154
Importing Data into the Hex Editor . . . 154
Inserting Data in the Hex Editor . . . 155
Appending Data in the Hex Editor . . . 155
Editing Data in the Hex Editor . . . 156
Reverting Changes in the Hex Editor . . . 158
Deleting Data in the Hex Editor . . . 158
Moving the Cursor in the Hex Editor . . . 159
Exporting Data from the Hex Editor . . . 159
Simulating a Policy . . . 160
Simulating Policies with Java Extensions . . . 164
Simulating Policies with Referenced Directories . . . 165

9 Storing Information in Resource Objects 167

Generic Resource Objects . . . 167
Creating a Generic Resource Object . . . 167
Using a Generic Resource Object . . . 168
Mapping Table Objects . . . 168
ECMAScript Objects . . . 168
Credential Application Objects . . . 169
Credential Repository Objects . . . 169
Package Objects . . . 169
DS Objects . . . 169
Package Prompts . . . 170
Filters . . . 170
Library Objects . . . 170
Creating Library Objects . . . 171
Adding Policies to the Library Objects . . . 171
Using Policies in the Library Objects . . . 172

10 Using ECMAScript in Policies 173

Creating an ECMAScript Object . . . 173
Using the ECMAScript Editor . . . 174
Main Scripting Area . . . 174
Expression Builder . . . 177
Functions and Variables . . . 179
Error Display . . . 180

Shell Area . . . 181
Examples of ECMAScripts with Policies . . . 183
DirXML Script Policy Calling an ECMAScript Function . . . 184
XSLT Policy Calling an ECMAScript Function at the Driver Level . . . 185
XSLT Policy Calling an ECMAScript Function in the Style Sheet . . . 186

11 Conditions 189

If Association . . . 190
If Attribute . . . 192
If Class Name . . . 195
If Destination Attribute . . . 198
If Destination DN . . . 201
If Entitlement . . . 203
If Global Configuration Value . . . 206
If Local Variable . . . 208
If Named Password . . . 212
If Operation . . . 213
If Operation Attribute . . . 216
If Operation Property . . . 220
If Password . . . 222
If Source Attribute . . . 225
If Source DN . . . 228
If XML Attribute . . . 230
If XPath Expression . . . 232

12 Actions 235

Add Association . . . 238
Add Destination Attribute Value . . . 239
Add Destination Object . . . 241
Add Resource . . . 243
Add Role . . . 245
Add Source Attribute Value . . . 247
Add Source Object . . . 248
Append XML Element . . . 249
Append XML Text . . . 251
Break . . . 253
Clear Destination Attribute Value . . . 254
Clear Operation Property . . . 255
Clear Source Attribute Value . . . 256
Clear SSO Credential . . . 257
Clone By XPath Expressions . . . 258
Clone Operation Attribute . . . 259
Create Resource . . . 261
Create Role . . . 264
Delete Destination Object . . . 266
Delete Source Object . . . 267
Delete Resource . . . 268
Delete Role . . . 269

Find Matching Object 270
For Each 273
Generate Event 274
If 277
If Operation Entitlement 279
Implement Entitlement 281
Invoke REST Endpoint 282
Modify Resource 284
Modify Role 286
Move Destination Object 288
Move Source Object 290
Reformat Operation Attribute 291
Remove Association 293
Remove Destination Attribute Value 294
Remove Named Password 295
Remove Role 296
Remove Resource 298
Remove Source Attribute Value 300
Rename Destination Object 301
Rename Operation Attribute 302
Rename Source Object 303
Send Email 304
Send Email from Template 307
Set Default Attribute Value 310
Set Destination Attribute Value 312
Set Destination Password 314
Set Local Variable 315
Set Named Password 317
Set Operation Association 318
Set Operation Class Name 319
Set Operation Destination DN 320
Set Operation Property 321
Set Operation Source DN 322
Set Operation Template DN 323
Set Source Attribute Value 324
Set Source Password 326
Set SSO Credential 327
Set SSO Passphrase 328
Set XML Attribute 329
Start Workflow 330
Status 332
Strip Operation Attribute 333
Strip XPath Expression 334
Trace Message 335
Veto 337
Veto If Operation Attribute Not Available 338
While 339

13 Noun Tokens 341

Text . . . 343
Added Entitlement . . . 345
Association . . . 346
Attribute . . . 347
Character . . . 348
Class Name . . . 349
Destination Attribute . . . 350
Destination DN . . . 352
Destination Name . . . 354
Document . . . 355
Entitlement . . . 356
Generate Password . . . 357
Global Configuration Value . . . 358
JSON Array . . . 359
JSON Object . . . 360
Local Variable . . . 361
Named Password . . . 362
Operation . . . 364
Operation Attribute . . . 365
Operation Property . . . 367
Password . . . 368
Query . . . 369
Removed Attribute . . . 371
Removed Entitlement . . . 372
Resolve . . . 373
Source Attribute . . . 374
Source DN . . . 375
Source Name . . . 376
Time . . . 377
Unique Name . . . 378
Unmatched Source DN . . . 381
XPath . . . 382

14 Verb Tokens 383

Base64 Decode . . . 384
Base64 Encode . . . 385
Convert Time . . . 386
Escape Destination DN . . . 388
Escape Source DN . . . 389
Join . . . 390
Lowercase . . . 391
Map . . . 392
Parse DN . . . 394
Replace All . . . 396
Replace First . . . 397
Split . . . 399
Substring . . . 400

Uppercase 402
XML Parse 403
XML Serialize 404

About this Book and the Library

The *Policies in Designer Guide* provides detailed information on working with OpenText™ Identity Manager CE 24.4 (v4.10) policies using Designer.

Intended Audience

This guide is intended for OpenText Identity Manager administrators.

Other Information in the Library

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

1 Overview

Policies manage the data that is synchronizing between the Identity Vault and the remote data store. The policies are stored in the policy sets (see "Understanding Policy Components" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.) Designer provides a wide set of tools for defining and debugging policies to control how information flows from one system to another, and under what conditions.

This section also contains a detailed reference section to all of the elements in DirXML Script. For more information on DirXML Script, see DirXML Script DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

Policies

As part of understanding how policies work, it is important to understand the components of policies.

- Policies are made up of rules.
- A rule is a set of conditions (see Chapter 11, "Conditions," on page 189) that must be met before a defined action (see Chapter 12, "Actions," on page 235) occurs.
- Actions can have dynamic arguments that derive from tokens that are expanded at runtime.
- Tokens are broken up into two classifications: nouns and verbs.
  - Noun tokens (see Chapter 13, "Noun Tokens," on page 341) expand to values that are derived from the current operation, the source or destination data stores, or some external source.
  - Verb tokens (see Chapter 14, "Verb Tokens," on page 383) modify the concatenated results of other tokens that are subordinate to them.
- Regular expressions (see "Regular Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide* and XPath 1.0 expressions (see "XPath 1.0 Expressions" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*) are commonly used in the rules to create the desired results for the policies.
- A policy operates on an XDS document and its primary purpose is to examine and modify that document.
- An operation is any element in the XDS document that is a child of the input element and the output element. The elements are part of `nds.dtd`; for more information, see NDS DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.
- An operation usually represents an event, a command, or a status.

- The policy is applied separately to each operation. As the policy is applied to each operation in turn, that operation becomes the current operation. Each rule is applied sequentially to the current operation. All of the rules are applied to the current operation unless an action is executed by a prior rule that causes subsequent rules to no longer be applied.
- A policy can also get additional context from outside of the document and cause side effects that are not reflected in the result document.

2 Managing Policies with the Policy Builder

The Policy Builder is a complete graphical interface for creating and managing the policies that define the exchange of data between connected systems.

- "Accessing the Policy Builder" on page 17
- "Using the Policy Builder" on page 19
- "Creating a Policy" on page 20
- "Creating a Rule" on page 24
- "Creating an Argument" on page 29
- "Variable Selector" on page 31
- "Editing a Policy" on page 35
- "Viewing the Policy in XML" on page 39

Accessing the Policy Builder

There are two different Policy Builders included in Designer: one that works with the new policy features for OpenText Identity Manager 3.5 and newer, and an older one that does not support these features. The Policy Builder version is determined by the version of OpenText Identity Manager. To set the version of OpenText Identity Manager:

1. Open a project in Designer.
2. Click the **Outline** tab, then select the **Show Model Outline** icon .
3. Right-click the server object, then click **Properties**.
4. Select the appropriate **OpenText Identity Manager Version**.

When the OpenText Identity Manager version is set to 3.5 or newer, the new Policy Builder is available. If the version is set to anything older than 3.5, the old Policy Builder is available.

The Policy Builder can be accessed from the Model Outline view, from the Policy Flow view, or from a policy set.

- "Model Outline View" on page 17
- "Policy Flow View" on page 18
- "Policy Set" on page 18

Model Outline View

1. Open a project in Designer.
2. Click the Outline view, then select the **Show Model Outline** icon .
3. Double-click a policy listed in the Model Outline view or right-click and select **Edit**.

Policy Flow View

1 Open a project in Designer.

2 Select the **Outline** tab, then select the **Show Policy Flow** icon.

3 Double-click a policy in the Policy Flow view.

You can also right-click in the Policy Flow view, select **Edit Policy**, then select the policy you want to edit.

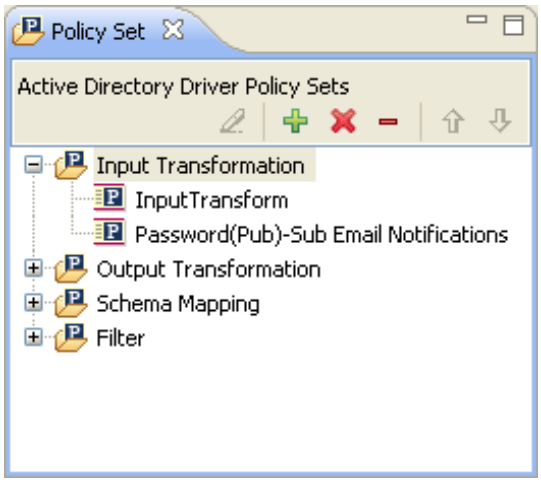
Policy Set

1 Open a project in Designer.

2 Click the Outline view, then select the **Show Model Outline** icon.

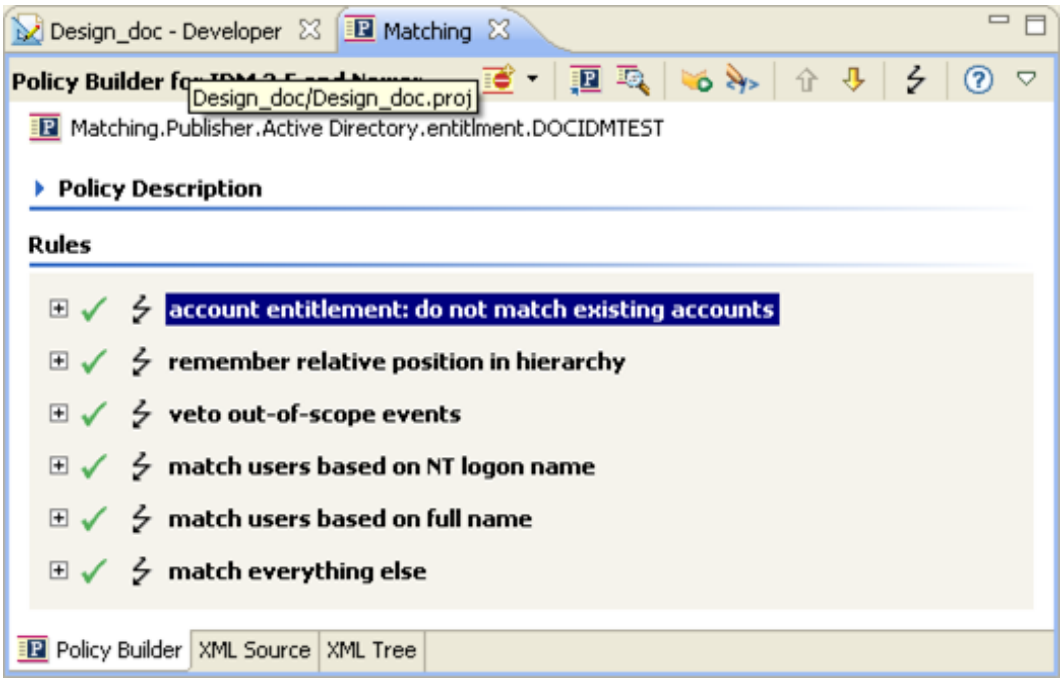
3 Select the policy in the policy set, then click **Edit the policy**.

You can also right-click the policy in the policy set, then click **Edit**.



To see all of the information in the Policy Builder window without scrolling, double-click the policy tab so the Policy Builder fills the entire window. To minimize the window, double-click the policy tab.

Figure 2-1 *Policy Builder Full Screen*



For information on using the Policy Builder, see "Using the Policy Builder" on page 19.

Using the Policy Builder

The Policy Builder enables you to add, view, and delete the rules that make up a policy. You can also use it to import and save policies and rules, and manage XML namespaces. The Policy Builder contains the "Action Builder" on page 41 and the "Condition Builder" on page 48.

The following tips describe how to perform some common Policy Builder tasks:

Table 2-1 *Common Policy Builder Tasks*

Tasks	Description
Disable	Disables a policy, rule, condition, or action.
Enable	Enables a policy, rule, condition, or action.
Disable Trace	Disables tracing on a rule, condition, or action.
Enable Trace	Enables tracing on a rule, condition, or action.
in the tool bar	Enables DirXML Script tracing on the policy.
Edit	Edits the name of a rule or edits the description of a rule.
Delete	Deletes a rule or a policy.

Tasks	Description
Browse	Browses a list of values to use when populating a field.
Add a rule	Adds a new rule or a predefined rule.
Import	Imports a policy from a file.
Save to File	Saves a policy to a file.
Deploy	Deploys a policy to the Identity Vault.
Compare	Compares the policy in the Policy Builder to an existing policy in the Identity Vault.
Policy Simulator	Launches the Policy Simulator and tests the policies in the Policy Builder.
Edit Namespace	Adds multiple XML namespaces to the rule or policy.
XPath Builder	Launches the XPath Builder to create XPath expressions.
Expand	Expands all of the rules in a policy.
Collapse	Collapses all of the rules in a policy.
Move up	Moves a rule up in the policy.
Move down	Moves a rule down in the policy.
Save	Click the save icon in the tool bar, click **File > Save**, or press Ctrl+S to save your work.
Policy Description	Adds a comment to a policy or rule. Comments are stored directly in the policy or rule, and can be as long as necessary.

Creating a Policy

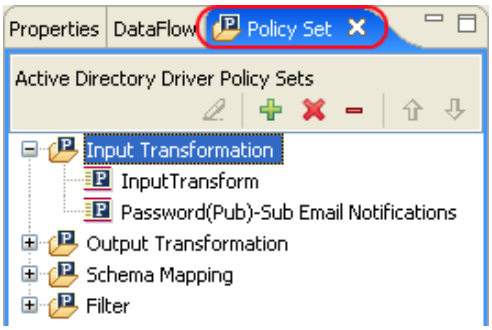
A policy sends data to the connected systems. A policy is created through the policy set.

- "Accessing the Policy Set" on page 21
- "Using the Policy Set" on page 21
- "Using the Add Policy Wizard" on page 22

Accessing the Policy Set

1 Select a driver object from the **Outline** view in an open project.

2 Select the **Policy Set** tab.



Using the Policy Set

The policy set contains a toolbar and a list of policies.

The policy list displays all the policies contained in the selected policy set. During a transformation, the policies within the list are executed from top to bottom. The toolbar contains buttons and a drop-down menu that you can use to manage policies displayed in the list, including, editing, adding, deleting, renaming, and changing the processing order of the policies.

Policy Set Toolbar

The policy set displays a copy of the policy. The buttons on the toolbar are enabled or disabled depending upon the item you have selected. The different icons are described below.

Table 2-2 *Policy Set Toolbar*

Operation	Description
Edit the policy	Launches the Policy Builder.
Create or add a new policy to the Policy Set	Launches the Add Policy Wizard.
Remove and delete the selected policy	Deletes the policy from the project.
Remove the selected policy from the Policy Set, but do not delete it	Removes the policy from the selected policy set object but doesn't delete the policy.
Move the policy up the policy chain	Moves the policy up in the processing order.
Move the policy down the policy chain	Moves the policy down in the processing order.

Keyboard Support

You can move through the policy set with keystrokes as well as using the mouse. The supported keystrokes are listed below.

Table 2-3 Keyboard Support

Keystroke	Description
Up-arrow	Moves the selected policy up in the processing order.
Down-arrow	Moves the selected policy down in the processing order.
Delete	Deletes the policy from the project.
Minus	Removes the policy from the selected policy set, but does not delete it.
Plus	Launches the Add Policy Wizard.
Ctrl+Z	Undoes the last operation.
Ctrl+Y	Redoes the last operation.

Using the Add Policy Wizard

The Add Policy Wizard launches when you click the **Create or add a new policy to the Policy Set** icon in the toolbar. The Add Policy Wizard enables you to do the following:

- "Creating a Policy" on page 22
- "Copying a Policy" on page 23
- "Linking to a Policy" on page 24

To launch the Add Policy Wizard:

1. Select a driver in the **Outline** view.
2. Select a policy set item in the policy set, then click **Create or add a new policy to the Policy Set** .

Creating a Policy

1. In the Add Policy Wizard, select **Create a new policy**, then click **Next**.

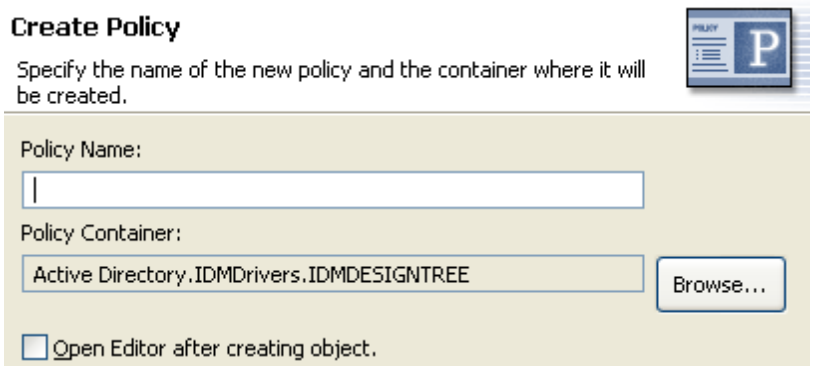
   You can also add a policy by right-clicking a policy set in the Policy Flow view, selecting **Add Policy**, then selecting how to create the policy:

   - **DirXML Script**
   - **XSLT**
   - **Link To Existing**
   - **Copy Existing**
   - **Schema Map** (Only displayed, if the Schema Map policy set is selected.)

   NOTE: DirXML Script and XSLT methods apply `is-sensitive` attribute on the XDS nodes to hide values of sensitive attributes such as passwords in the trace file. For a sample trace output showing the use of this attribute, see Working with is-sensitive Attribute in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

2 In the Create Policy dialog box, specify a policy name, then click **Next**.

Select Open Editor after creating object to automatically launch the Policy Builder after creating the new policy.



Accept the default container, or browse to and select the Driver, Publisher, or Subscriber object where you want the policy to be created.

If a policy is not reused by multiple drivers, you typically create that policy under the driver or channel that is using it.

This decision depends on how you want to organize the policies. By default, policies are placed under the container object that is selected in the **Outline** tab when the Add Policy Wizard is launched.

For example, if you move to a Publisher object in the **Outline** tab and then add a policy to a policy set, the policy defaults to the Publisher container.

You can change this setting if you want to create policies in a different container. For example, you can set up a policy library, put all of the common policies under this driver, and then simply reference the policies from the other drivers. That way, the policy is common. If you need to change a policy, you need to do it only once.

3 In the Select Type dialog box, select the type of policy you want to implement, then click **Finish**.

The policy type defaults to **DirXML Script**. You can select **XSLT**, if you don't want to use DirXML Script.

If you create a Schema Map policy set, an additional option is available for **Schema Map**.

The new policy appears in the expanded policy set.

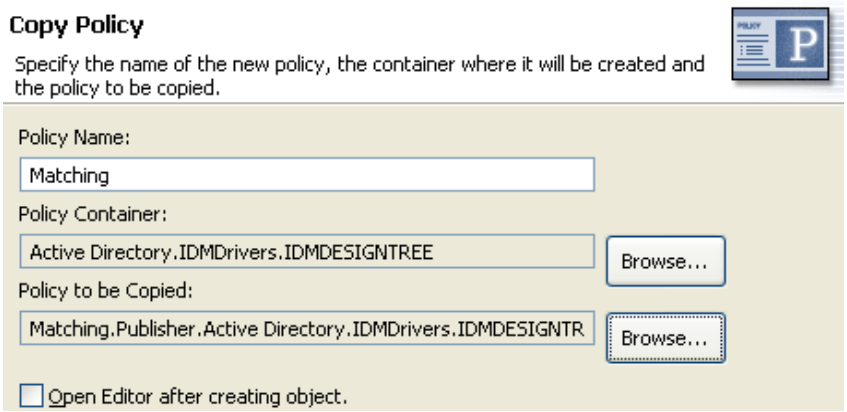
Copying a Policy

1 In the Add Policy Wizard, select **Copy a policy**, then click **Next**.

2 In the Create Policy dialog box, provide the necessary policy information, then click **OK**.

- Specify a name for the new policy

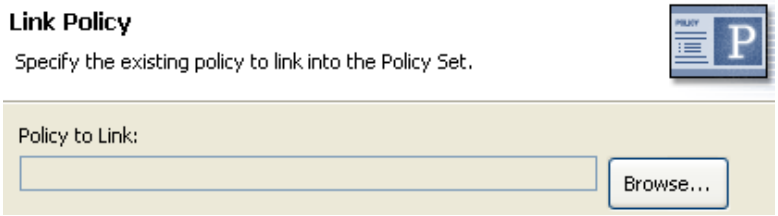
- Accept the default container, or browse to and select the Driver, Publisher, or Subscriber object where you want the policy to be created.
- Browse to and select the policy you want to copy, then click **Finish**.



Linking to a Policy

1 In the Add Policy Wizard, select **Link a policy**, then click **Next**.

2 In the Link Policy dialog box, click **Browse** to launch the model browser.



3 Browse to and select the Policy object you want to link into the policy set, then click **OK**.

Linking a policy into a policy set doesn't create a new Policy object. Instead, it adds a reference to an existing policy. This reference can be to any existing policy within the current Identity Vault. It doesn't need to be contained within the current Driver object, but the policy type must be valid for the policy set that it is being linked to. For example, you can't link a Schema Map policy into an Input policy set.

Linking a policy into a policy set is not permitted when viewing all policies.

4 Click **Finish** to link to the selected policy.

Creating a Rule

A rule is a set of conditions that must be met before a defined action occurs. Rules are created from condition groups, conditions, and actions.

Rules can be created in four different ways:

- "Creating a New Rule" on page 25
- "Using Predefined Rules" on page 27

- "Including an Existing Rule" on page 28
- "Importing a Policy From an XML File" on page 28

Creating a New Rule

When you create a rule, you create condition groups, conditions, and actions. Each rule is composed of conditions, actions, and arguments. For more information, click the Help icon when creating each item. The help files contain a definition and an example of the item being used.

- "Creating a Rule" on page 25
- "Creating a Conditional Group" on page 26
- "Creating a Condition" on page 27
- "Creating an Action" on page 27

Creating a Rule

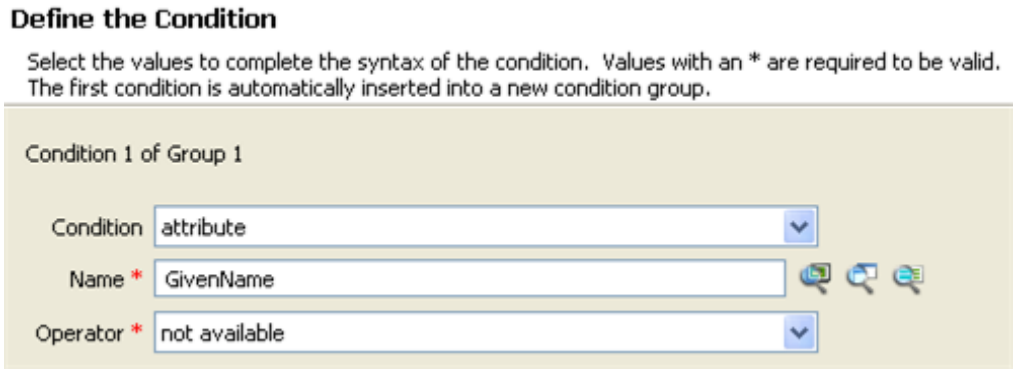
Policy Builder includes a wizard to step you through the process of creating a rule.

NOTE: On any of the wizard dialog boxes, you can click **Finish** to exit the wizard and create a rule with the details you have specified to that point.

---

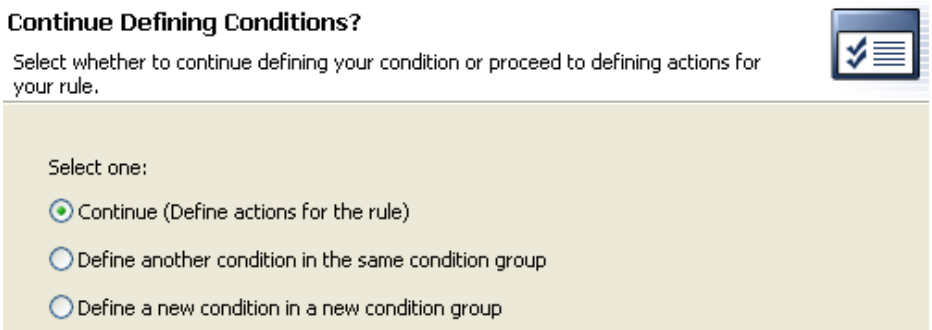
1. In Policy Builder toolbar, click **Rule** .
2. In the Name and Describe Rule dialog box, specify the name of the rule, then click **Next**.
3. In the Select the Condition Structure dialog box, select the rule's condition structure, then click **Next**.

   You can choose **OR Conditions, AND Groups** or **AND Conditions, OR Groups**.
4. In the Define the Condition dialog box, select the condition you want, specify the appropriate information, then click **Next**.



   The icons next to the **Name** field let you browse the Identity Vault schema, the connected application schema, or use the Variable Selector to select the appropriate information.
5. In the Continue Defining Conditions dialog box, select the appropriate option, then click **Next**.

   If desired, you can define additional conditions or condition groups before proceeding. For this example, there is only one condition.



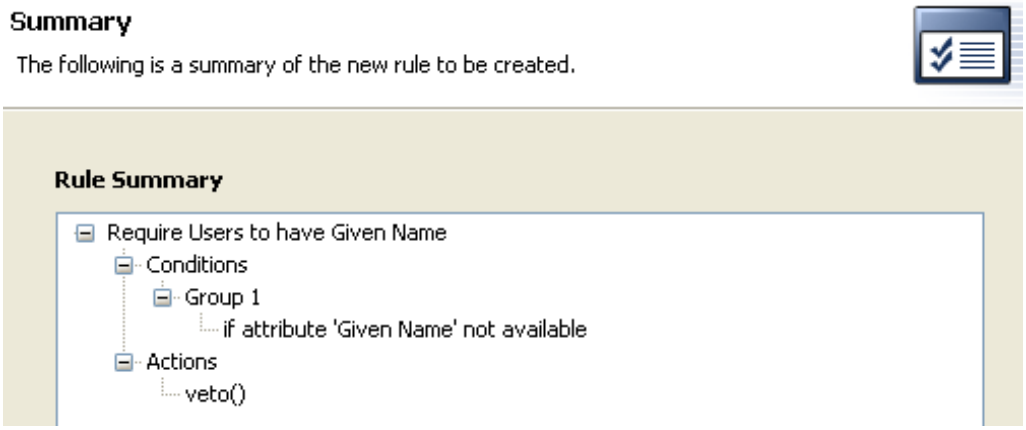
6 In the Define the Action dialog box, select the action that you want, then click **Next**.

7 In the Continue Defining Actions dialog box, select the appropriate option, then click **Next**.

If desired, you can define additional actions before proceeding. For this example, there is only one action.

8 In the Summary page, click **Finish** to create the rule.

You can expand or collapse the view of the rule by clicking the plus or minus sign.

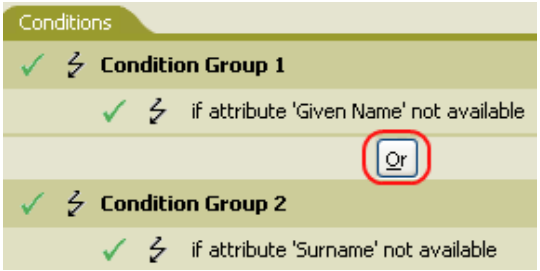


Creating a Conditional Group

1 In the Policy Builder, right-click the **Conditions** tab then click **Append Condition Group**.

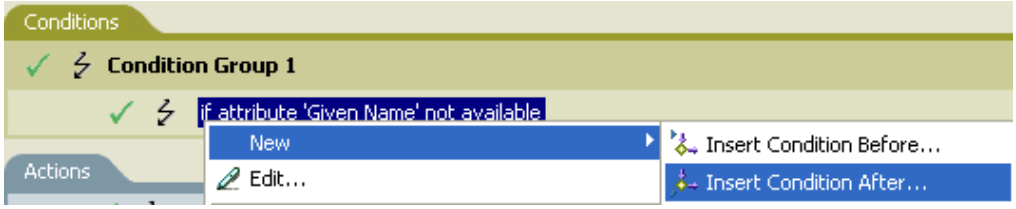
You can also right-click the name of the **Condition Group**, then click **New > Insert Condition Group Before** or **Insert Condition Group After**.

Change the condition for the Condition Groups by clicking the **And/Or** icon.

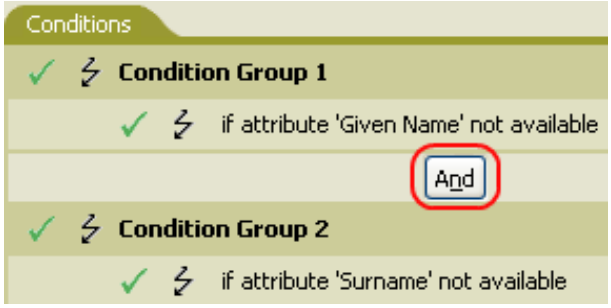


Creating a Condition

1 Right-click the condition, then click **New > Insert Condition Before** or **Insert Condition After**.

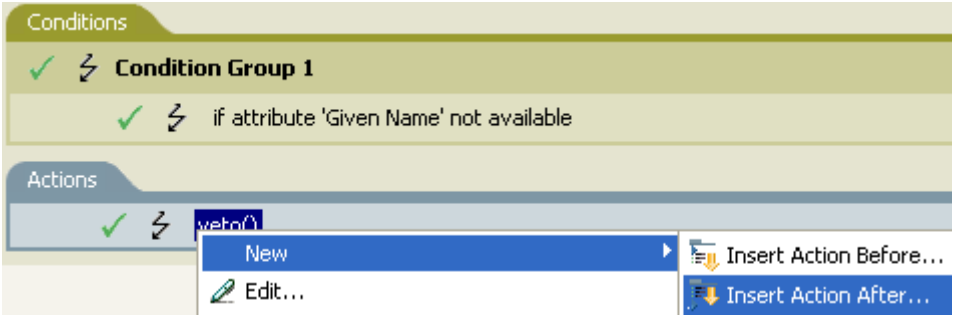


You can change the condition by clicking the **And/Or** icon.



Creating an Action

1 Right-click the action, then click **New > Insert Action Before** or **Insert Action After**.

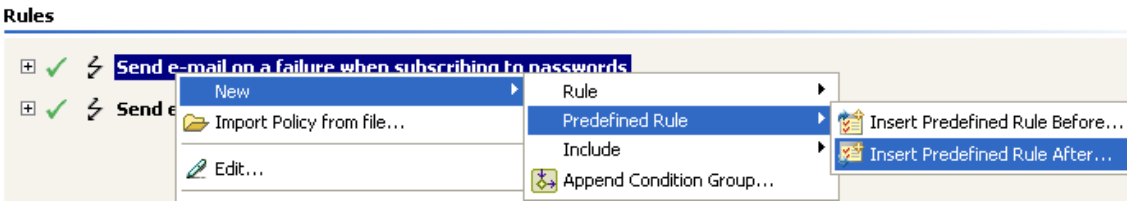


Using Predefined Rules

Designer includes a list of predefined rules. You can import and use these rules as well as create your own rules.

1 Right-click in the Policy Builder and select **New > Predefined Rules > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

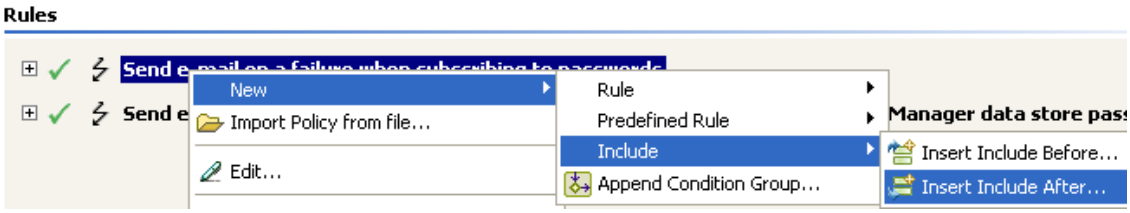
See Chapter 7, “Using Predefined Rules,” on page 107 for more information.



Including an Existing Rule

Designer allows you to include the rules from another policy.

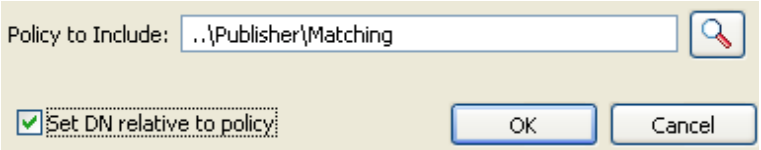
1 Right-click in the Policy Builder and click **New > Include > Insert Include Before** or **Insert Include After**.



2 Click the Browse icon .

3 Browse to the policy you want to include, then click **OK**.

4 The field is now populated with the path to the policy. Click **OK**.

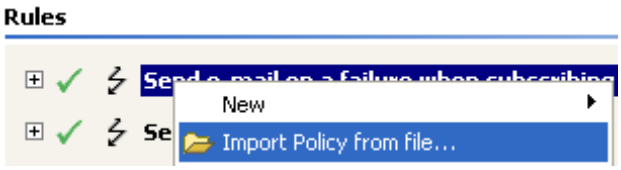


The rule is a link to the original rule. You cannot edit the rule in this location. Access the original rule to make changes.

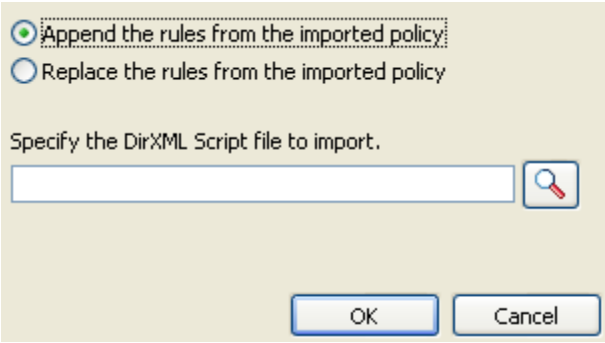
Importing a Policy From an XML File

Rules and policies can be saved as XML files. If you have a file that contains a rule or a policy you want to use, the Policy Builder allows you to import the file.

1 In the Policy Builder, right-click and select **Import Policy from file**.



2 Select one of the two options: **Append the rules from the imported policy** or **Replace the rules from the imported policy**.



3 Click the browse icon and select the file that contains the policy, then click **Open**.

4 Click **OK**.

Creating an Argument

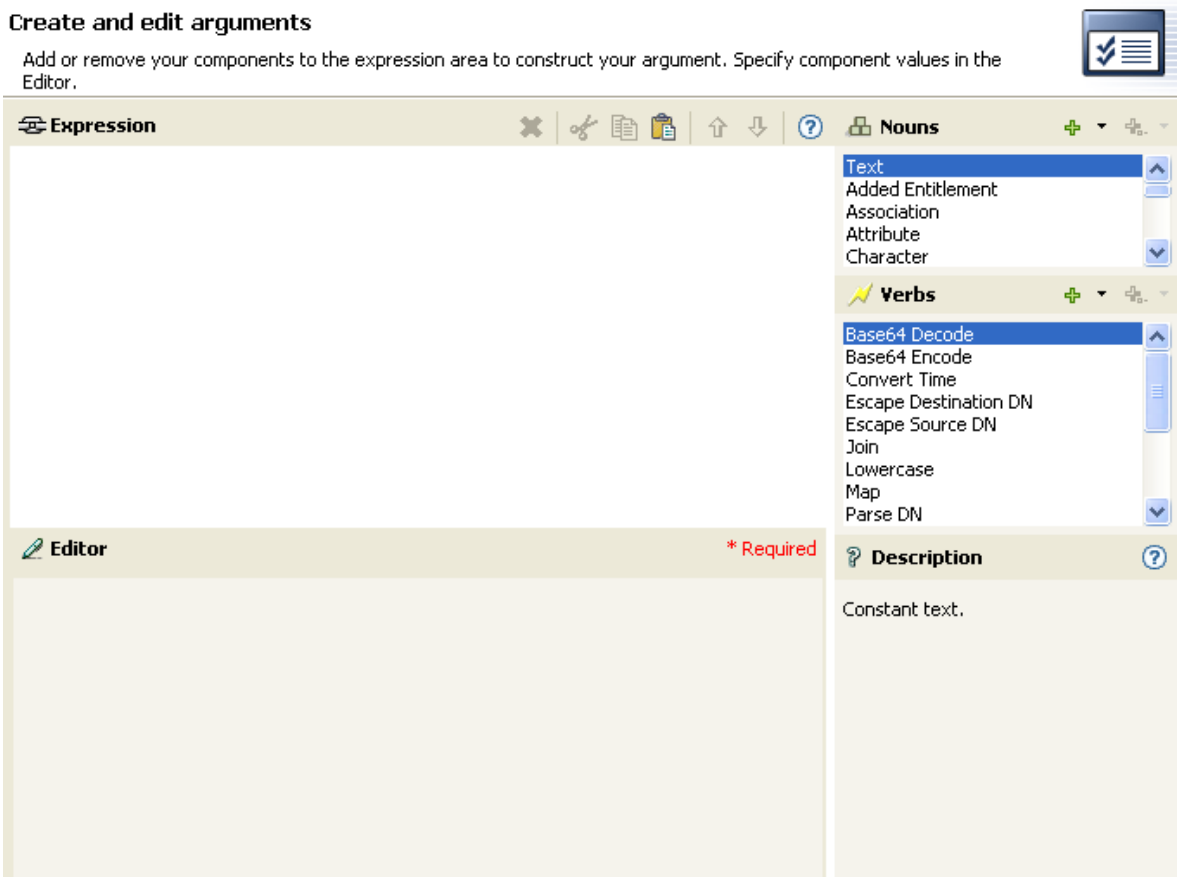
The Argument Builder provides a dynamic graphical interface that enables you to construct complex argument expressions for use within the Policy Builder. To access the Argument Builder, see "Argument Builder" on page 43.

Arguments are dynamically used by actions and are derived from tokens that are expanded at run time.

Tokens are broken up into two classifications: nouns and verbs. Noun tokens expand to values that are derived from the current operation, the source or destination data stores, or some external source. Verb tokens modify the results of other tokens that are subordinate to them.

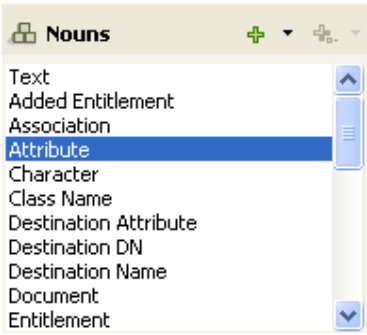
To define an expression, select one or more noun tokens (values, objects, variables, etc.), and combine them with verb tokens (substring, escape, uppercase, and lowercase) to construct arguments. Multiple tokens are combined to construct complex arguments.

Figure 2-2 Argument Builder



For example, if you want the argument set to an attribute value, you select the attribute noun, then select the attribute name:

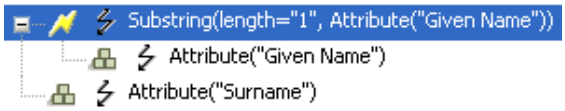
1 Double-click **Attribute** in the list of noun tokens to add it to the **Expression** pane.



2 Browse to and select the attribute name in the **Editor** field.

You can browse the Identity Vault schema or the connected application schema.

If you only want a portion of this attribute, you can combine the attribute token with the substring token. The expression displays a substring length of 1 for the Given Name attribute combined with the entire Surname attribute.

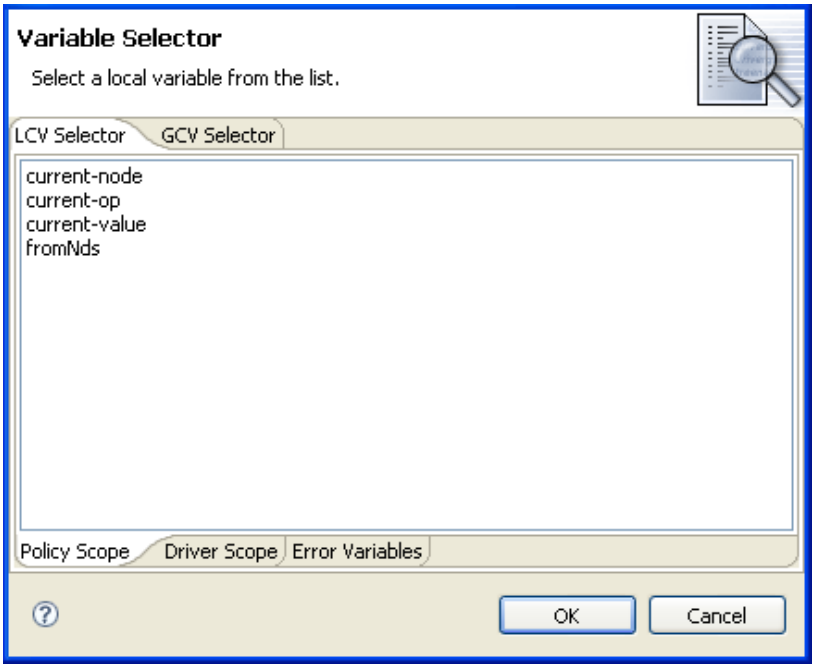


After you add a noun or verb, you can provide values in the editor, then immediately add another noun or verb. You do not need to refresh the Expression pane to apply your changes; they appear when the next operation is performed.

See "Noun Tokens" on page 341 and "Verb Tokens" on page 383 for a detailed reference on the noun and verb tokens. See "Argument Builder" on page 43 for more information on the Argument Builder.

Variable Selector

The variable selector provides a list of variables that you can select and insert into conditions, actions, and tokens.



- "Dynamic Variable Expansion" on page 32
- "Accessing the Variable Selector From the Conditions Tab" on page 32
- "Accessing the Variable Selector From the Actions Tab" on page 33
- "Accessing the Variable Selector From the Argument Builder" on page 34
- "XPath Expressions" on page 35

Dynamic Variable Expansion

The variable selector allows for the use of dynamic variable expansion in conditions, actions, and tokens. It is used when the writer of the DirXML script doesn't know what value to enter during the design phase, and wants the value to be populated dynamically when the code is run (for local variables) or when the driver starts (for global variables). Dynamic variables are not used when the policy needs to refer directly to the variable itself. Instead, they are used when the policy needs to refer to the value of the variable.

Many actions support dynamic variable expansion in their attributes or content. Where supported, an embedded reference of the form *$variable-name$* is replaced with the value of the local variable with the given name. An embedded reference of the form **~variable-name~** is replaced with the value of the global variable name. *$variable-name$* and **~variable-name~** must be legal variable names. For information on what constitutes a legal XML name, see W3C Extensible Markup Language (XML) (http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-suggested-names).

If the given variable does not exist, the reference is replaced with the empty string. Where it is desirable to use a single $ and not have it interpreted as a variable reference, use an additional $ as an escape character (for example, You owe me $$100.00).

NOTE: If the global variable does not exist on the driver or driver set, the driver does not start.

Accessing the Variable Selector From the Conditions Tab

1. In the Policy Builder, double-click the **Conditions** tab.

   For instructions on accessing the Policy Builder, see "Accessing the Policy Builder" on page 17.
2. Select one of the following conditions:
   - If Attribute (page 192)
   - If Destination Attribute (page 198)
   - If Entitlement (page 203)
   - If Global Configuration Value (page 206)
   - If Local Variable (page 208)
   - If Named Password (page 212)
   - If Operation Attribute (page 216)
   - If Source Attribute (page 225)
3. Click the **Launch variable browser** icon next to the field where you want to insert a dynamic variable.
4. Select the variable, then click **OK**.

Or, for conditions that don't bring up the **Launch variable browser** icon:

1. Select one of the following operators:
   - Equal
   - Greater than
   - Less than

- Not equal
- Not greater than
- Not less than

2 Click the **Launch variable browser** icon next to the field where you want to insert the dynamic variable.

3 Select the variable, then click **OK**.

Accessing the Variable Selector From the Actions Tab

1 In the Policy Builder, double-click the **Actions** tab.

For instructions on accessing the Policy Builder, see "Accessing the Policy Builder" on page 17.

2 In the **Do** field, select one of the following options:

- Add Destination Attribute Value (page 239)
- Add Destination Object (page 241)
- Add Role (page 245)
- Add Source Attribute Value (page 247)
- Add Source Object (page 248)
- Append XML Element (page 249)
- Append XML Text (page 251)
- Clear Destination Attribute Value (page 254)
- Clear Source Attribute Value (page 256)
- Clear SSO Credential (page 257)
- Clone By XPath Expressions (page 258)
- Clone Operation Attribute (page 259)
- Delete Destination Object (page 266)
- Delete Role (page 269)
- Move Destination Object (page 288)
- Move Source Object (page 290)
- Reformat Operation Attribute (page 291)
- Remove Destination Attribute Value (page 294)
- Remove Role (page 296)
- Remove Source Attribute Value (page 300)
- Rename Destination Object (page 301)
- Rename Operation Attribute (page 302)
- Rename Source Object (page 303)
- Send Email from Template (page 307)
- Set Default Attribute Value (page 310)
- Set Destination Attribute Value (page 312)
- Set Destination Password (page 314)

- Set Local Variable (page 315)Set Source Attribute Value (page 324)
- Set Source Password (page 326)
- Set SSO Credential (page 327)
- Set SSO Passphrase (page 328)
- Set XML Attribute (page 329)
- Start Workflow (page 330)
- Strip Operation Attribute (page 333)
- Strip XPath Expression (page 334)
- Veto If Operation Attribute Not Available (page 338)

3 Click the **Launch variable browser** icon next to the field where you want to insert the dynamic variable.

4 Select the variable, then click **OK**.

Accessing the Variable Selector From the Argument Builder

1 In the Argument Builder, select one of the following noun tokens from the **Nouns** section:

- Text (page 343)
- Added Entitlement (page 345)
- Attribute (page 347)
- Destination Attribute (page 350)
- Entitlement (page 356)
- Generate Password (page 357)
- Global Configuration Value (page 358)
- Local Variable (page 361)
- Named Password (page 362)
- Operation Attribute (page 365)
- Query (page 369)
- Removed Attribute (page 371)
- Removed Entitlement (page 372)
- Source Attribute (page 374)
- Time (page 377)
- Unique Name (page 378)
- XPath (page 382)

Or, select one of the following verb tokens from the **Verbs** section:

- Convert Time (page 386)
- Map (page 392)

2 Click the **Launch variable browser** icon next to the field where you want to insert the dynamic variable.

3 Select the variable, then click **OK**.

XPath Expressions

Instead of using the DirXMLScript engine to perform the variable expansion, as is the case with most variable expansion, XPath uses built in XPath functionality and the XSLT processor to do the variable expansion.

For conditions, actions, and tokens that contain XPath expressions, a single $ sign at the beginning of the policy denotes a dynamic variable, which is displayed in the **Value** field. This is also true for the XPath token in the Argument Builder, and for all actions that contain XPath. This is because in order to maintain valid XPath, only one $ sign can be used.

The following procedure gives an example of using the variable selector with XPath expressions:

1 In the Policy Builder, click the **Actions** tab.

2 In the **Do** field, select the **clone by XPath expressions** option.

3 After the **Specify source XPath expression** field, click the **Launch variable browser** icon .

4 Select an item and click **OK**.

Only one $ sign is displayed before the policy.

Editing a Policy

The Policy Builder allows you to create and edit policies. You can drag and drop rules, conditions and actions. For additional operations, access the Policy Builder toolbar. To display a context menu, right-click an item.

- "Actions and Menu Items in the Policy Builder" on page 35
- "Keyboard Support" on page 37
- "Renaming a Policy" on page 37
- "Saving Your Work" on page 38
- "Policy Description" on page 38

Actions and Menu Items in the Policy Builder

The table contains a list of the different actions and menu items in the Policy Builder.

Table 2-4 *Policy Builder Actions and Menu Items*

Operation	Description
Collapse All	Collapses all expanded rules.

Operation	Description
Compare Deployed Policy	Compares the policy in the Policy Builder to an existing policy in the Identity Vault.
Copy	Copies the selected item to the Clipboard.
Copy and drop	Select the item, press Ctrl, then drag the item.
Cut	Cuts the selected item and copies it to the Clipboard.
Delete	Deletes the selected item.
Deploy Policy	Deploys the policy into the Identity Vault.
Disable	Displays a rule, condition, or action as disabled.
Disable Trace	Disables trace on the rule.
DirXML Script Tracing	Enables DirXML Script Tracing on the policy.
Drag and drop	Enables you to select an item, then relocate it. Select the item, then drag it to the new location.
Edit	Enables you to edit the selected item. To open the Rule Builder, select a rule, then click Edit.
Enable	Displays a rule, condition, or action as enabled.
Enable Trace	Enables tracing on the rule.
Expand All	Expands all the rules so that you can view the conditions and actions of each rule.
Import Policy from file	Imports a policy from the file system and appends it to the policy, or replaces all the rules of the policy.
Launch Policy Simulator	Launches the Policy Simulator.
Move and drop	Enables you to select and move an item. Select the item, then drag it.
Move down	Moves the item down in the list of policies.
Move up	Moves the item up in the list of policies.
New > Append Condition Group	Creates a new condition group after a selected item.
New > Include > Insert Include Before or Insert Include After	Creates a new Include before or after the selected item.
New > Predefined Rule > Insert Predefined Rule Before or Insert Predefined Rule After	Inserts a predefined rule before or after the selected item.
New > Rule > Insert Rule Before or Insert Rule After	Creates a new rule before or after the selected item.
Paste	Pastes the contents of the Clipboard after the selected item.

Operation	Description
Preferences	Enables you to change how the information is displayed.
Redo	Redoes the previous action.
Select	Click any item to select it.
Undo	Undoes the previous action.

Keyboard Support

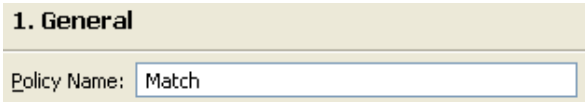
You can move through the Policy Builder with keystrokes as well as using the mouse. The supported keystrokes are listed below.

Table 2-5 *Keyboard Support in the Policy Builder*

Keystroke	Description
Ctrl+C	Copies the selected item into the Clipboard.
Ctrl+X	Cuts the selected item and adds it to the Clipboard.
Ctrl+V	Pastes the contents of the Clipboard after the selected item.
Delete	Deletes the selected Item.
Left-arrow	Collapses a rule node.
Right-arrow	Expands a rule node.
Up-arrow	Navigates up.
Down-arrow	Navigates down.
Ctrl+Z	Undo
Ctrl+Y	Redo

Renaming a Policy

1. In the Outline view, select the policy you want to rename.
2. Right-click and select **Properties**.
3. Change the name of the policy in the **Policy Name** field.



4. Click **OK**.

Saving Your Work

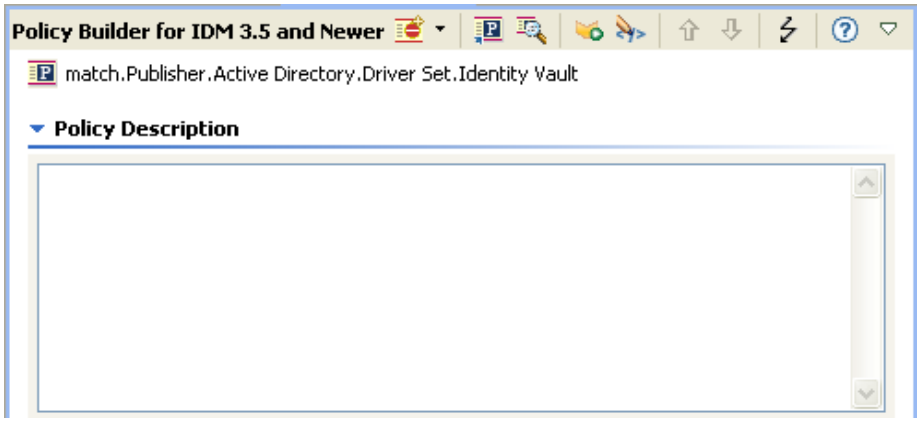
Do one of the following:

- From the main menu, click **File** > **Save** (or **Save All**).
- Close the editor by clicking the **X** in the editor's tab.
- Select **Close** from the main menu's file menu.
- Press Ctrl+S.

Policy Description

The description fields provide a place to add notes about the functionality of the policy. You can add a description for the policy and you can add a description for the rule.

1. In the Policy Builder, click **Policy Description**.



2. Provide a description of the policy.
3. Save the policy by pressing Ctrl+S.

To add a description to a rule:

1. Double-click the name of the rule.

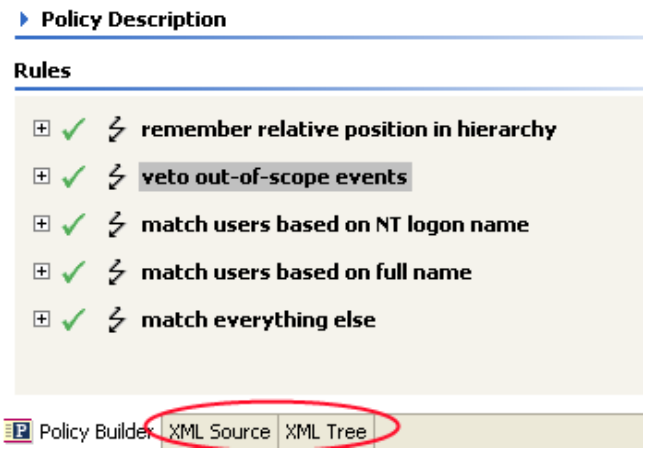
2 Specify a description of the rule in the **Description** field.

3 Save the rule by pressing Ctrl+S.

Viewing the Policy in XML

Designer enables you to view, edit, and validate the XML by using an XML editor. Click the **XML Source** or **XML Tree** tabs to access the XML editor. For more information about the XML editor, see "The NetIQ XML Editor" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Figure 2-3 *View Policy in XML*



3 Using Additional Builders and Editors

Although you define most arguments in the Argument Builder, there are several more builders and editors that are used by the Condition editor and Action editor in the Policy Builder. Each builder can recursively call anyone of the builders in the following list:

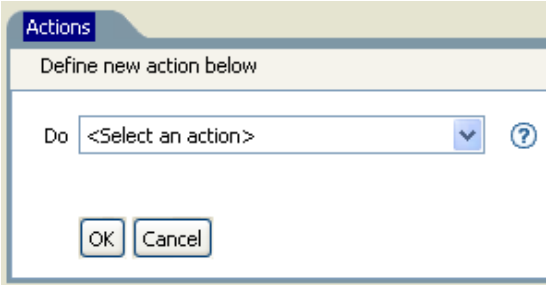
- "Action Builder" on page 41
- "Actions Builder" on page 42
- "Argument Builder" on page 43
- "Condition Builder" on page 48
- "Conditions Builder" on page 50
- "Match Attribute Builder" on page 51
- "Action Argument Component Builder" on page 53
- "Argument Value List Builder" on page 54
- "Named String Builder" on page 55
- "Condition Argument Component Builder" on page 56
- "Pattern Builder" on page 57
- "String Builder" on page 58
- "XPath Builder" on page 58
- "Mapping Table Editor" on page 58
- "Global Configuration Value Definition Editor" on page 64
- "Namespace Editor" on page 65
- "Local Variable Selector" on page 67

Action Builder

The Action Builder enables you to add, view, and delete the actions that make up a rule. Actions can also contain other actions.

Creating an Action

1. In the Policy Builder, create a new rule or edit an existing rule.
2. Double-click the **Actions** tab to launch the Action Builder.



3 Select the desired action from the drop-down list, then click **OK**.

Additional Options for the Action Builder

There are additional options in the action builder to manage the actions. Right-click the action to see the additional options.

Table 3-1 *Action Builder Additional Options*

Option	Description
New > Insert Action Before	Adds a new action before the current action.
New > Insert Action After	Adds a new action after the current action.
Edit	Launches the Action Builder.
Move up	Moves the selected action up in the order of execution.
Move down	Moves the selected action down in the order of execution.
Cut	Cuts the selected action and adds it to the clipboard.
Copy	Copies the action to the clipboard.
Paste	Pastes the action that is in the clipboard to the desired location in the Action Builder.
Delete	Deletes the selected action.
Undo	Undoes the prior action.
Redo	Redoes the prior action.
Preferences	Allows you to set default functionality in the Policy Builder.

Actions Builder

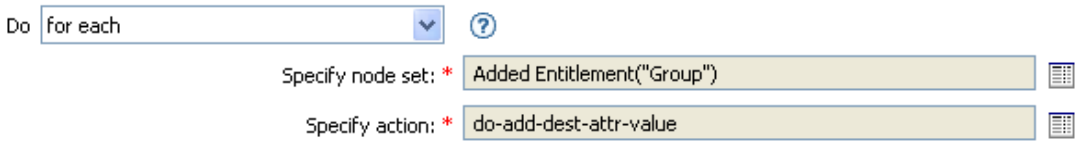
The Actions Builder allows you to create an action inside of another action. To launch the Actions Builder, select one of the following actions, then click the **Edit the actions** icon .

- For Each (page 273)
- Implement Entitlement (page 281)

- If (page 277)
- While (page 339)

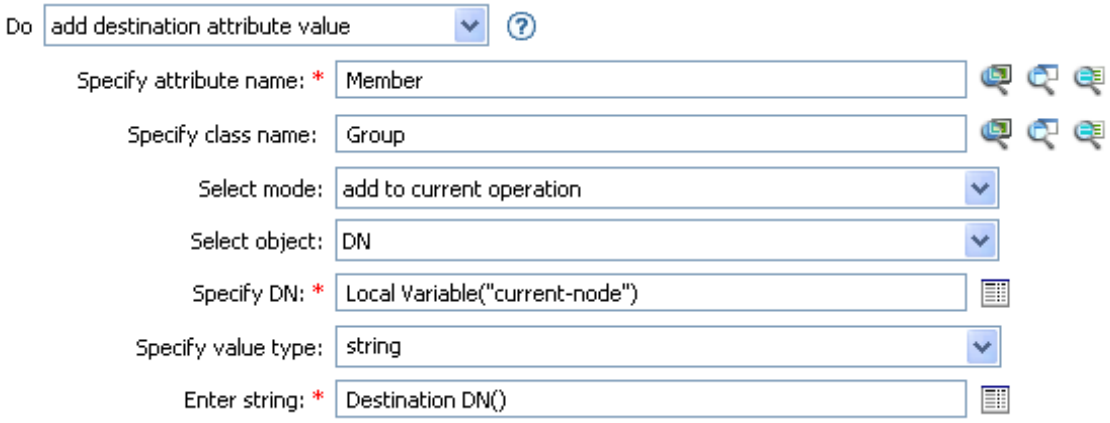
In the following example the add destination attribute value action is performed for each Group entitlement that is being added in the current operation.

Figure 3-1 *For Each Action*



To define the action of the add destination attribute value, click the **Edit the actions** icon. This launches the Actions Builder. In the Actions Builder, you define the desired action. In the following example, the member attribute is added to the destination object for each added Group entitlement.

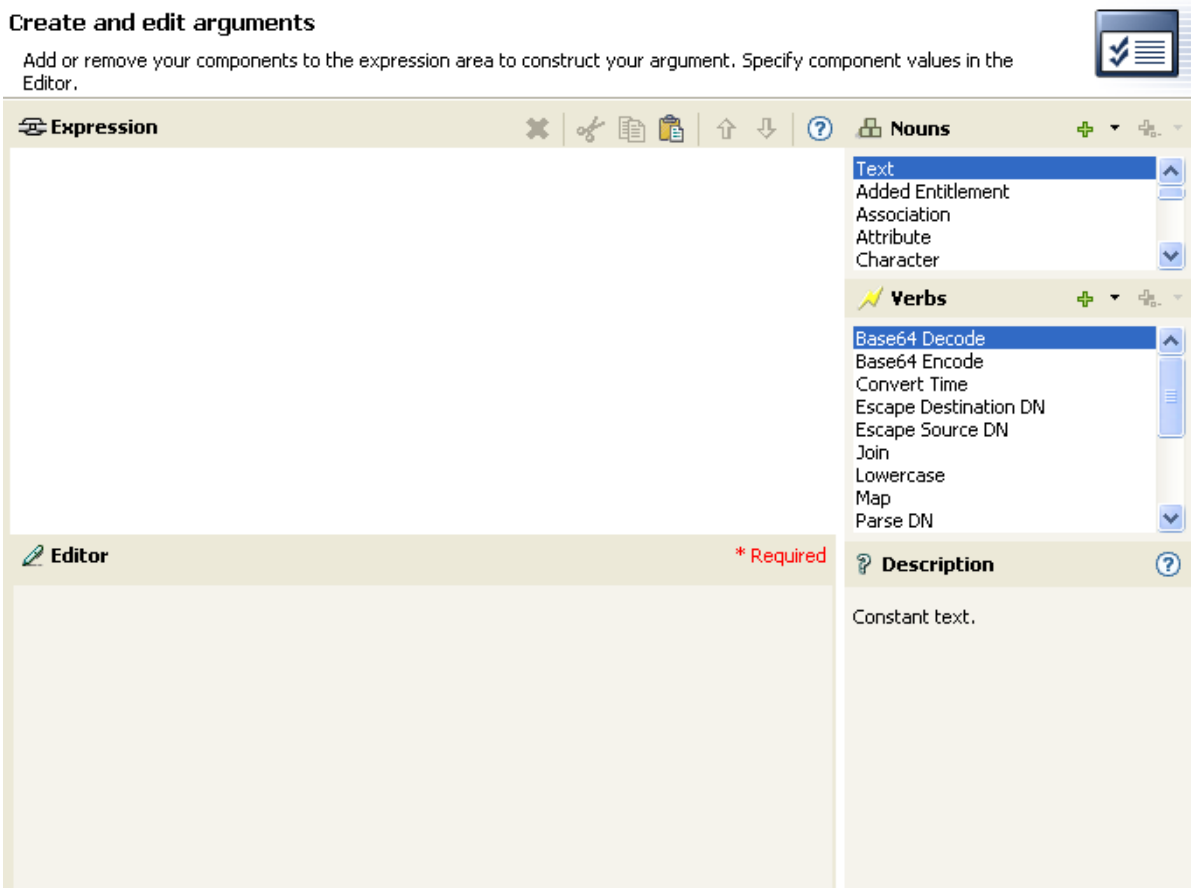
Figure 3-2 *Actions Builder*



Argument Builder

The Argument Builder provides a dynamic graphical interface that enables you to construct complex argument expressions for use within Rule Builder.

Figure 3-3 Argument Builder



The Argument Builder consists of six separate sections:

Nouns: Contains a list of all of the available noun tokens. Double-click a noun token to add it to the **Expression** pane. See "Noun Tokens" on page 341 for more information.

Verbs: Contains a list of all of the available verb tokens. Double-click a verb token to add it to the **Expression** pane. See "Verb Tokens" on page 383 for more information.

Description: Contains a brief description of the selected noun or verb token. Click the **Help** icon to launch additional help.

Expression: Contains the argument that is being built. Multiple noun and verb tokens can be added to a single argument. Tokens can be arranged in different orders through the **Expression** pane.

Editor: Provide the values for the nouns and the verbs in the **Editor** pane.

Toolbar: Allows you to manipulate the noun and verb tokens. See Table 3-2 for a list of all of the options in the toolbar.

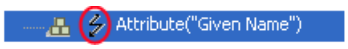
Table 3-2 Argument Builder Toolbar Options

Option	Description
Delete	Deletes the selected token.

Option	Description
Cut	Cuts the selected token to the Clipboard.
Copy	Copies the selected token to the Clipboard.
Paste	Pastes the token from the Clipboard into the Argument Builder.
Move Up	Moves the selected token up.
Move Down	Moves the selected token down.
Help	Launches the help.
Append noun	Appends a noun token to the end of the argument as a sibling token.
Insert noun	Inserts a noun token into the argument.
Append noun to child token list	Appends a noun token as a child token instead of as a sibling token.
Insert noun at beginning of child token list	Inserts a noun token into the argument as the first child in the token list instead of as a sibling token.
Append verb	Appends a verb token to the end of the argument as a sibling token.
Insert verb	Inserts a verb token into the argument.
Append verb to child token list	Appends a verb token as a child token instead of as a sibling token.
Insert verb at beginning of child token list	Inserts a verb token into the argument as the first child in the token list instead of as a sibling token.

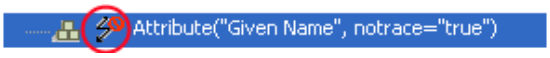
You can select to trace each token or disable the tracing of the token in the Argument Builder. To disable tracing:

1 Click the trace icon to disable tracing.



To enable tracing:

1 Click the disable trace icon to enable tracing.



- "Launching the Argument Builder" on page 46
- "Argument Builder Example" on page 47

Launching the Argument Builder

To launch the Argument Builder, select one of the following actions, then click the **Edit the arguments** icon .

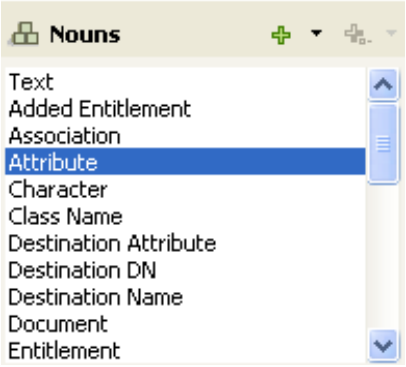
- Add Association (page 238)
- Add Destination Attribute Value (page 239)
- Add Destination Object (page 241)
- Add Source Attribute Value (page 247)
- Append XML Text (page 251)
- Clear Destination Attribute Value (page 254) (when the selected object is DN or Association)
- Clear Source Attribute Value (page 256) (when the selected object is DN or Association)
- Delete Destination Object (page 266) (when the selected object is DN or Association)
- Delete Role (page 269) (when the selected object is DN or Association)
- Find Matching Object (page 270)
- For Each (page 273)
- Move Destination Object (page 288)
- Move Source Object (page 290)
- Reformat Operation Attribute (page 291)
- Remove Association (page 293)
- Remove Destination Attribute Value (page 294)
- Remove Source Attribute Value (page 300)
- Rename Destination Object (page 301) (when the selected object is DN or Association and Enter String)
- Rename Source Object (page 303) (when the selected object is DN or Association and Enter String)
- Set Destination Attribute Value (page 312) (when the selected object is DN or Association and Enter Value Type is not structured)
- Set Destination Password (page 314)
- Set Local Variable (page 315)
- Set Operation Association (page 318)
- Set Operation Class Name (page 319)
- Set Operation Destination DN (page 320)
- Set Operation Property (page 321)
- Set Operation Source DN (page 322)
- Set Operation Template DN (page 323)
- Set Source Attribute Value (page 324)
- Set Source Password (page 326)
- Set XML Attribute (page 329)

- Status (page 332)
- Trace Message (page 335)

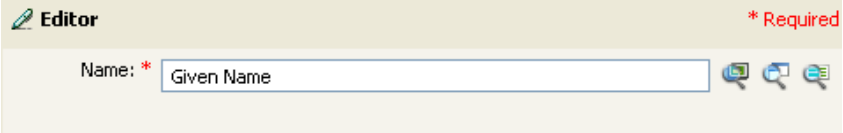
Argument Builder Example

The following example creates an argument for a username from the first letter of the first name and the entire last name:

1 Double-click **Attribute** from the list of nouns.

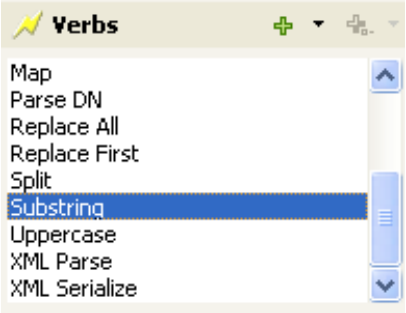


2 Specify or select the Given Name attribute.

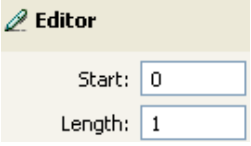


You can browse the Identity Vault attributes, the application attributes, or launch the variable browser. For more information on the variable browser, see "Variable Selector" on page 31.

3 Double-click **Substring** from the list of verbs.



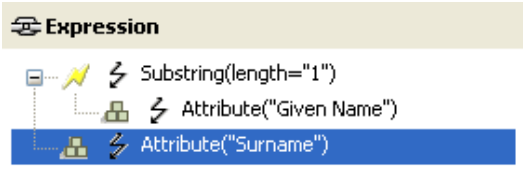
4 Type 1 in the **Length** field.



5 Select the **Given Name** attribute, then click the **Move Down** icon.

6 Double-click **Attribute** from the list of nouns.

7 In the **Name** field, specify or browse to the **Surname** attribute.



The argument takes the first character of the Given Name attribute and adds it to the Surname attribute to build the desired value.

8 Click **Finish** to save the argument.

Condition Builder

The Condition Builder enables you to add, view, and delete the conditions that make up a rule. A condition contains one or more conditions and one or more condition groups. The condition groups contain two different condition structures, which define the logic of condition groups. The two condition structures are:

- OR Conditions, AND Groups
- AND Conditions, OR Groups

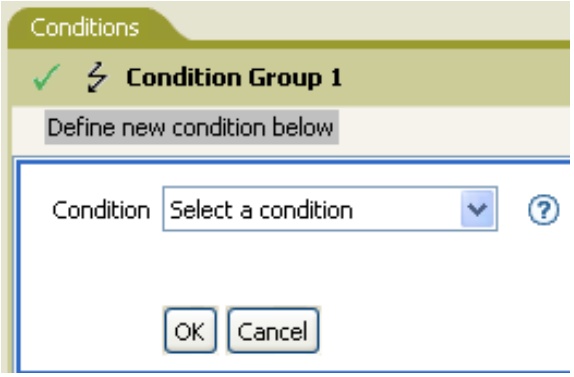
To create and customize a condition, see the following sections:

- "Creating a Condition" on page 48
- "Additional Options for the Condition Builder" on page 49

Creating a Condition

1 In the Policy Builder, create a new rule or edit an existing rule.

2 Double-click the **Conditions** tab to launch the Condition Builder.



3 Select the desired condition from the drop-down list, then click **OK**.

Additional Options for the Condition Builder

There are additional options in the condition builder to manage the conditions. Right-click the condition to see the additional options.

Table 3-3 *Condition Builder Options*

Option	Description
New > Insert Condition Before	Adds a condition before the current condition.
New > Insert Condition After	Adds a condition after the current condition.
Edit	Launches the Condition Builder.
Move up	Moves the selected condition up in the order of execution.
Move down	Moves the selected condition down in the order of execution.
Cut	Cuts the select condition and adds it to the clipboard.
Copy	Copies the condition and adds it to the clipboard.
Paste	Pastes the condition that is in the clipboard in the desired location in the Condition Builder.
Delete	Deletes the selected condition.
Undo	Undoes the prior action in the Condition Builder.
Redo	Redoes the prior action in the Condition Builder.
Preferences	Allows you to set default functionality in the Policy Builder.

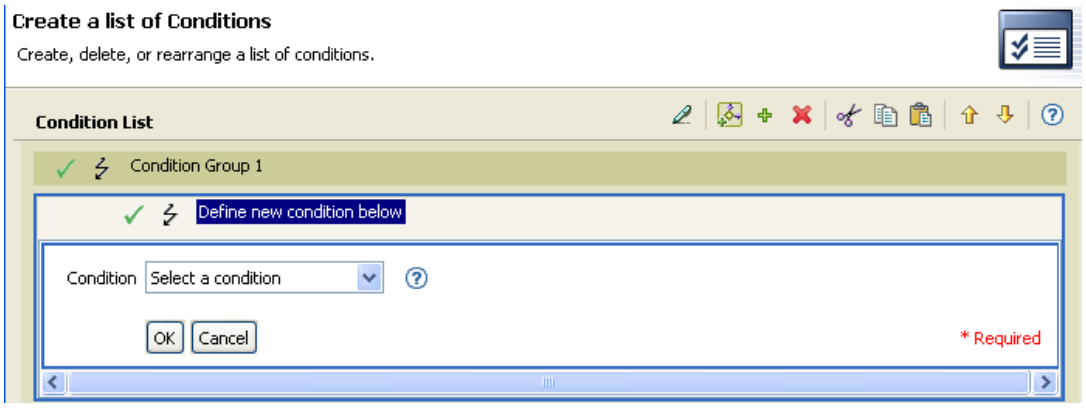
For additional information on the Condition Builder and the rules, see "Creating a Rule" on page 24.

Conditions Builder

The Conditions Builder allows you to create a condition inside of an action. To launch the Conditions Builder, select one of the following actions, then click the **Edit the actions** icon next to the **If conditions** field.

- If (page 277)
- While (page 339)

1 In the Conditions Builder, browse to and select the desired condition.



2 Define the condition, then click **OK**.

The Conditions Builder has additional options that the Condition Builder. Right-click the Conditions Builder.

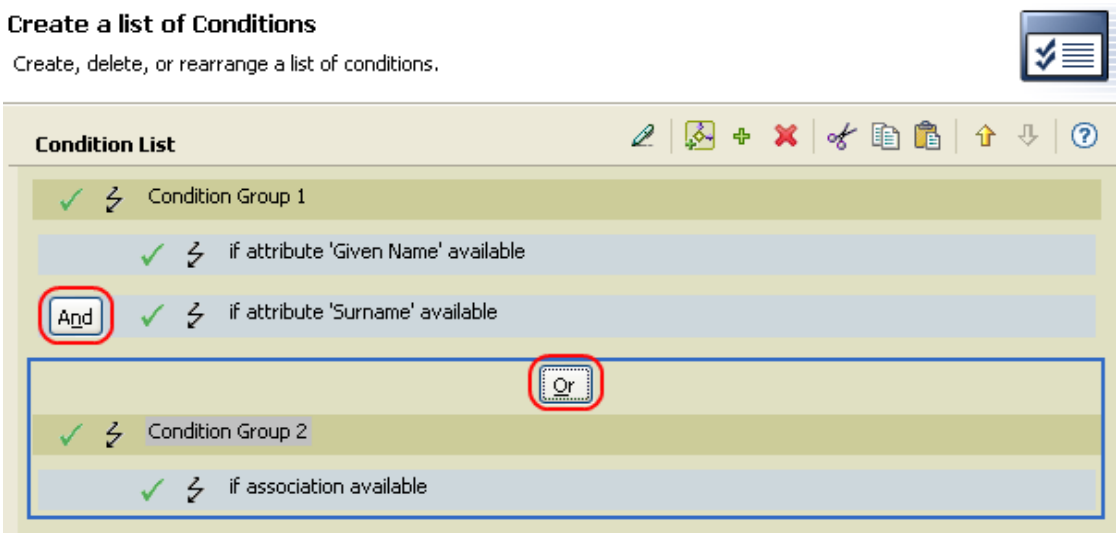
Table 3-4 *Conditions Builder Options*

Option	Description
New > Insert Condition Group Before	Adds a condition group before the selected condition group.
New > Insert Condition Group After	Adds a condition group after the selected condition group.
Append Conditions	Appends a condition in the condition group.
Expand All Conditions	Expands all conditions that are part of the selected condition group.
Collapse All Conditions	Collapses all conditions that are part of the selected condition group.
Move up	Moves the selected condition group up in the rule.
Move down	Moves the selected condition group down in the rule.
Cut	Cuts the selected condition group from the rule and adds it to the clipboard.

Option	Description
Copy	Copies the selected condition group and adds it to the clipboard.
Paste	Pastes the condition group from the clipboard into the Conditions Builder.
Delete	Deletes the selected condition or condition group.
Undo	Undoes the prior action in the Conditions Builder.
Redo	Redoes the prior action in the Condition Builder.
Preferences	Allows you to set default functionality in the Policy Builder.

If you have multiple conditions and conditions groups, the **And/Or** icons are tied together. If you change the **And/Or** icon for the condition groups, it is changed for the conditions as well.

Figure 3-4 *Conditions Builder And/Or Icons*

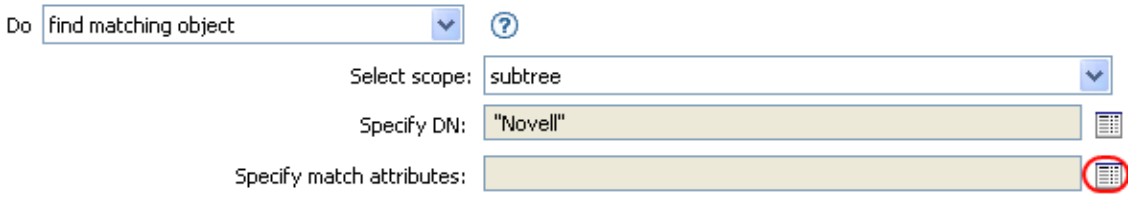


Match Attribute Builder

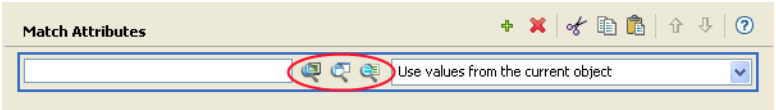
The Match Attribute Builder enables you to select attributes and values used by the Find Matching Object (page 270) action to determine if a matching object exists in a data store.

For example, if you wanted to match users based on a common name and a location:

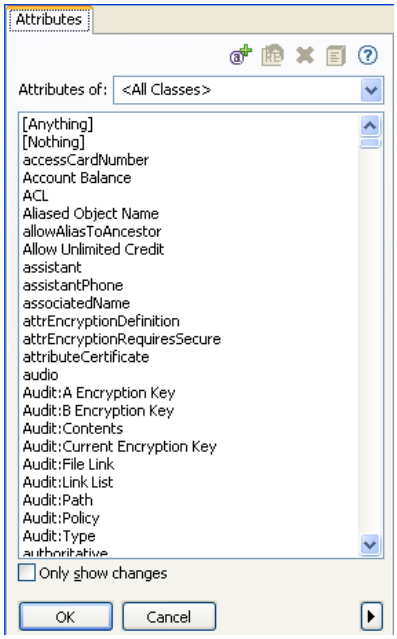
1. Select the action of **find matching object**.
2. Select the scope of the search for the matching objects. Select from **entry**, **subordinates**, or **subtree**.
3. Specify the DN of the starting point for the search.
4. Click the **Edit match attributes** icon to launch the Match Attribute Builder.



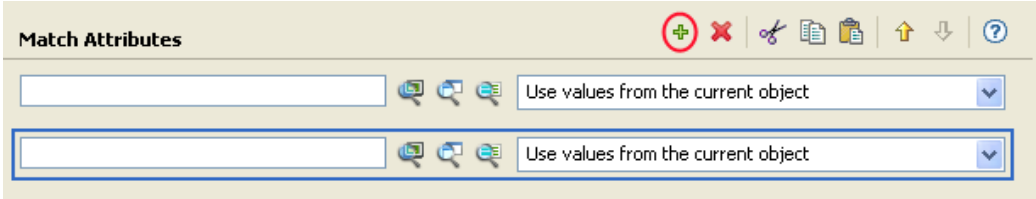
5 Click the **Browse the Identity Vault attributes** icon, the **Browse application attributes** icon, or the **Launch variable browser** icon. For more information on the **Launch variable browser** icon, see "Variable Selector" on page 31.



6 Browse to and select the desired attribute, then click **OK**.



If you want to add more than one attribute, click the **Append new item** icon to add another line.



You can browse the Identity Vault schema or the connected system schema.

7 Click **Finish**.

The Match Attribute Builder also allows you to specify another value, instead of using the value from the current object. For more information about value types, see **value** in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

To use another value:

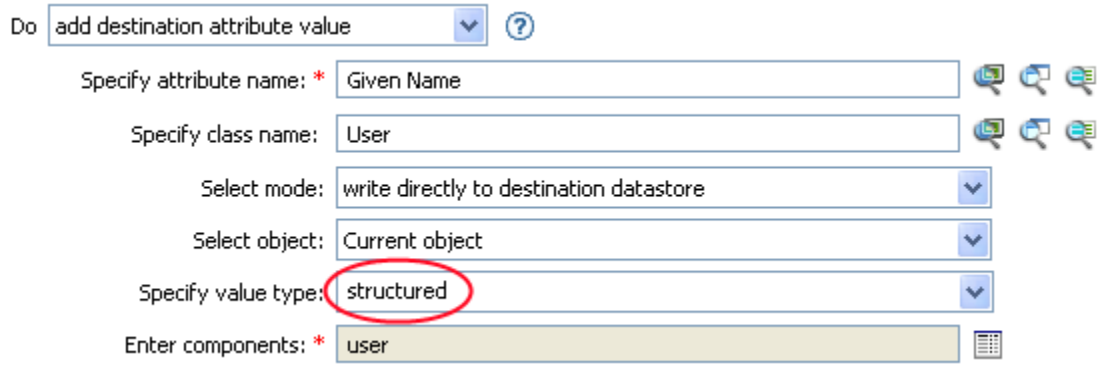
1. Launch the Match Attribute Builder, then select **Other Value** from the drop-down list.
2. Select the desired value type.
   - counter
   - dn
   - int
   - interval
   - octet
   - state
   - string
   - structured
   - teleNumber
   - time
3. Specify the value, then click **OK**.

Action Argument Component Builder

To launch the Action Argument Component Builder, select one of the following actions when the **Select Value Type** selection is **structured**, then click the **Edit the components** icon .

- Add Destination Attribute Value (page 239)
- Add Source Attribute Value (page 247)
- Reformat Operation Attribute (page 291)
- Remove Destination Attribute Value (page 294)
- Remove Source Attribute Value (page 300)
- Set Destination Attribute Value (page 312)
- Set Source Attribute Value (page 324)

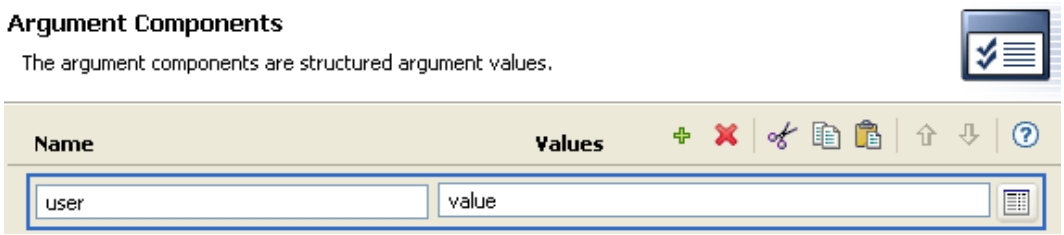
Figure 3-5 Add Destination Attribute Value Action



1 Make sure the value type is set to **structured**, then click the **Edit the components** icon.

2 Create the value of the action component.

You can type the value, or click the **Edit the arguments** icon to create the value in the Argument Builder.



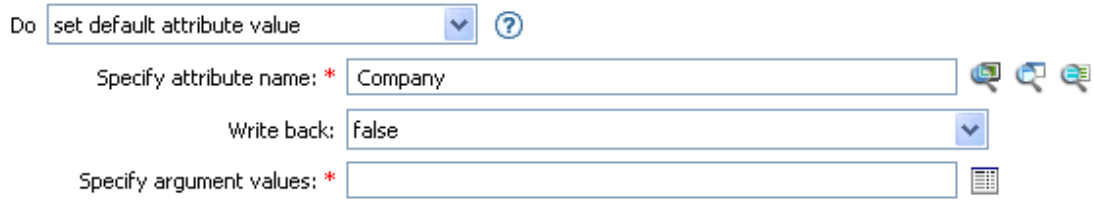
3 Click **Finish**.

Argument Value List Builder

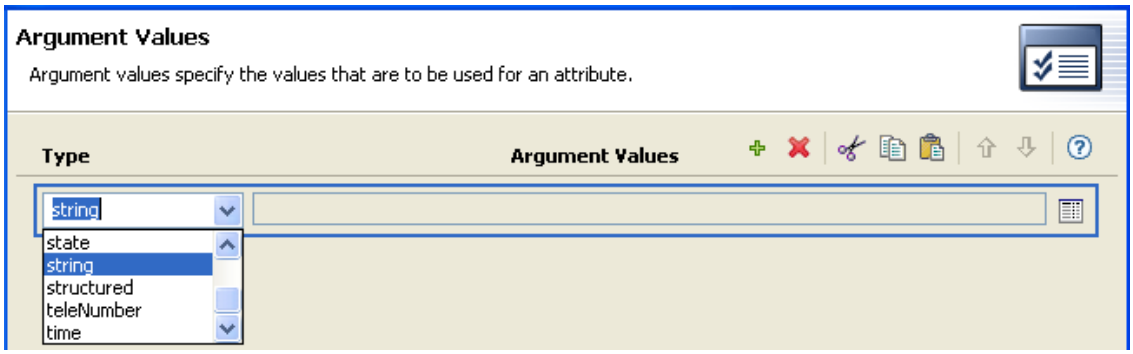
To launch the Argument Value List Builder, select the following action, then click the **Edit the arguments** icon.

- Set Default Attribute Value (page 310)

Figure 3-6 Set Default Attribute Value



1 Select the type of the value: **counter**, **dn**, **int**, **interval**, **octet**, **state**, **string**, **structured**, **teleNumber**, **time**.



2 Create the value of the list.

You can type the value, or click the **Edit the arguments** icon to create the value in the Argument Builder.

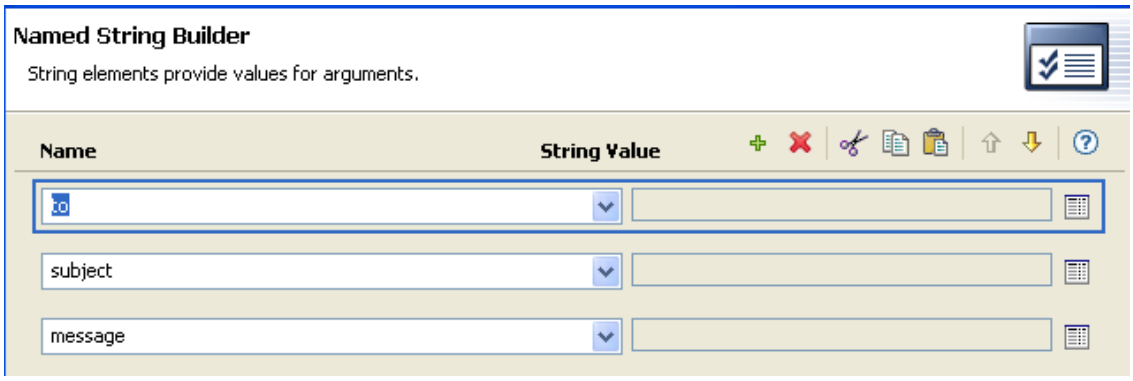
3 Click **Finish**.

Named String Builder

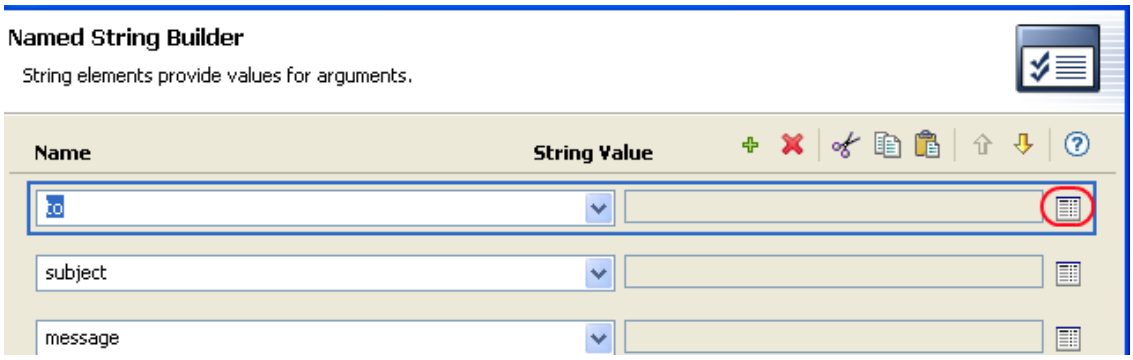
To launch the Named String Builder, select one of the following actions, then click the **Edit the strings** icon next to the **Strings** field.

- Add Role (page 245)
- Generate Event (page 274)
- Remove Role (page 296)
- Send Email (page 304)
- Send Email from Template (page 307)
- Start Workflow (page 330)

1 Select the name of the string from the drop-down list.



2 Create the value for the string by clicking the **Edit the arguments** icon to launch the Argument Builder.



3 Click **Finish**.

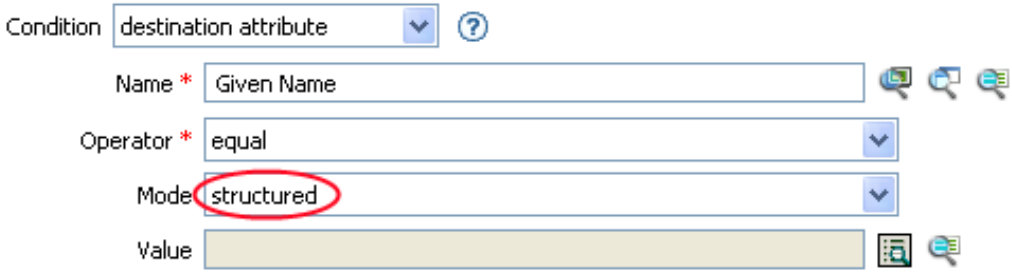
For a Send Email action, the named strings correspond to the elements of the e-mail. A complete list of possible values is contained in the help file corresponding to the action that launches the Named String Builder.

Condition Argument Component Builder

To launch the Condition Argument Component Builder, select one of the following conditions, then select the structured selection for **Mode** in order to see the **Launch ArgComponent Builder** icon.

- If Attribute (page 192)
- If Destination Attribute (page 198)
- If Operation Attribute (page 216)
- If Source Attribute (page 225)

Figure 3-7 *If Attribute mode*



1 Specify the name and value of the condition component.

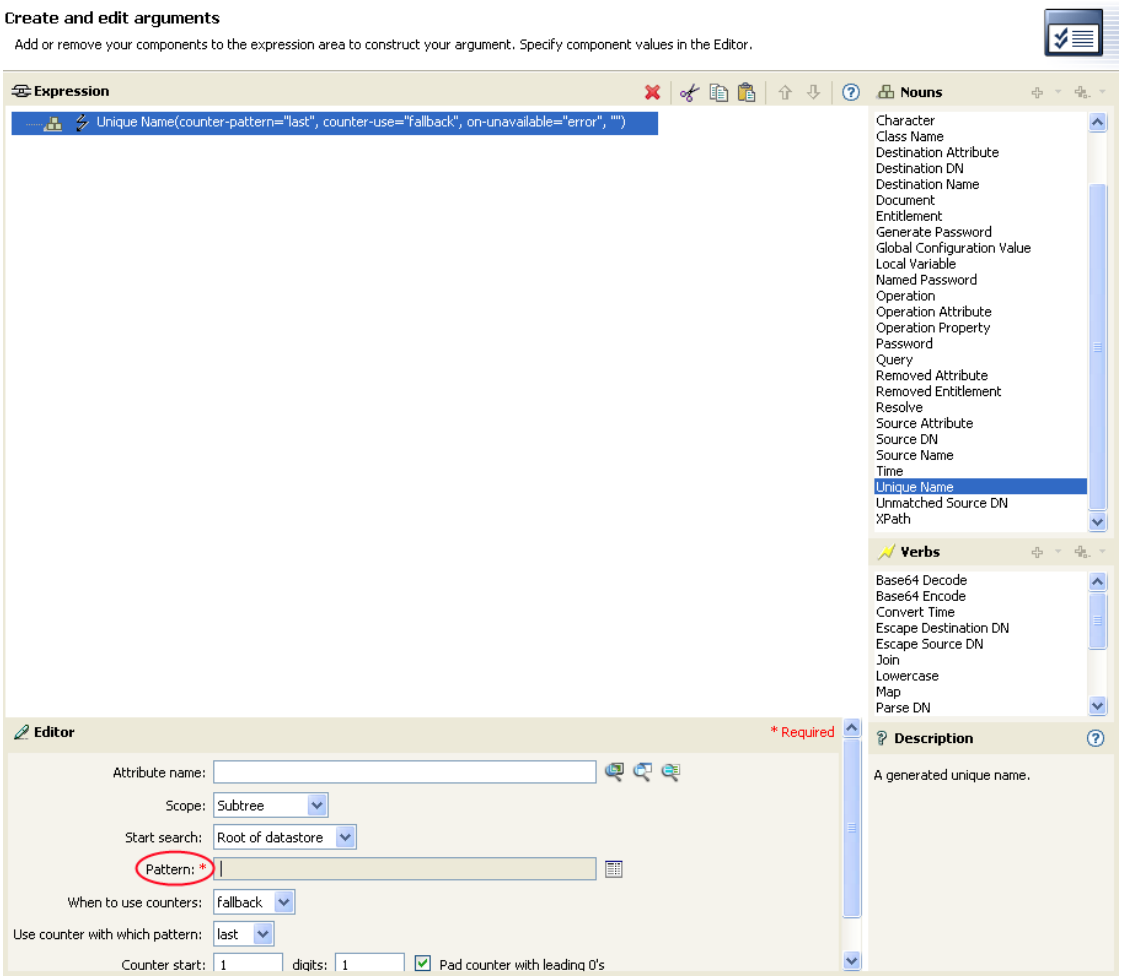
2 Click **Finish**.

Pattern Builder

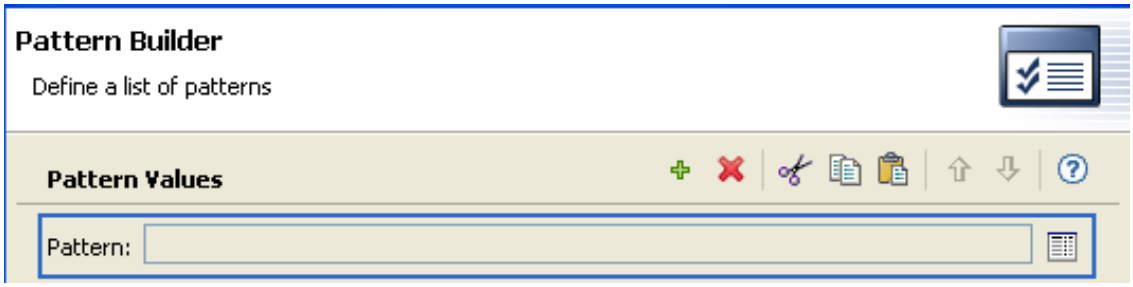
You can launch the Pattern Builder from the Argument Builder editor when the Unique Name (page 378) token is selected. The Argument Builder editor pane shows a **Pattern** field where you can click to launch the Pattern Builder.

For information on how to access the Argument Builder, see "Launching the Argument Builder" on page 46.

Figure 3-8 *Unique Name Token in the Argument Builder*



1 Click the **Edit patterns** icon to launch the Pattern Builder.



2 Specify the pattern or click the **Edit the arguments** icon to use the Argument Builder to create the pattern.

3 Click **Finish**.

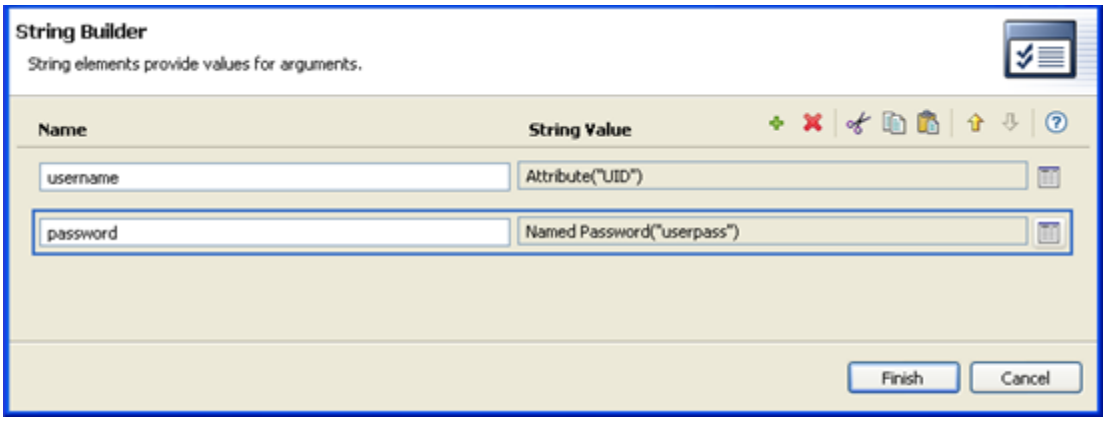
String Builder

The String Builder enables you to construct name/value pairs for use in certain actions, including Set SSO Credential and Clear SSO Credential.

To open String Builder, select the **Edit the Strings** icon next to the appropriate field when defining a new action or modifying an existing action. For example, The Set SSO Credential action contains a **Login Parameter Strings** field for necessary login parameter strings. String Builder allows you to create the appropriate strings.

In the String Builder, specify a name for each string you want to add to the action, then manually, or using the Argument Builder, create the appropriate string value.

Figure 3-9 *String Builder Example*



XPath Builder

The XPath Builder is a powerful tool that allows you to build and test an XPath expression against any XML document. See "Using the XPath Builder" on page 69 for more information.

Mapping Table Editor

The Mapping Table editor allows you to create, edit, and manage mapping table objects. A mapping table object is used by a policy to map a set of values to another set of corresponding values. After a mapping table object is created, the Map (page 392) token maps the results of the specified tokens from the values specified in the mapping table.

To use a mapping table object, the following steps must be completed:

1. "Creating a Mapping Table Object" on page 59
2. "Adding a Mapping Table Object to a Policy" on page 60

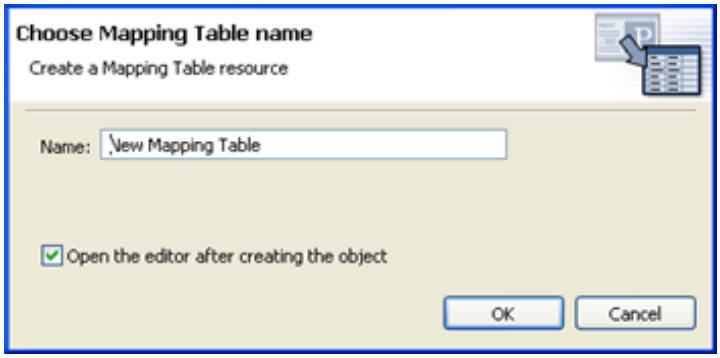
To edit a mapping table, see "Editing a Mapping Table Object" on page 62.

Creating a Mapping Table Object

A mapping table object can be created in a library, driver object, Publisher channel, or Subscriber channel.

1. In the Outline view, right-click the location to create the mapping table, then select **New > Mapping Table**.
2. Specify the name of the mapping table object, then click **OK**.

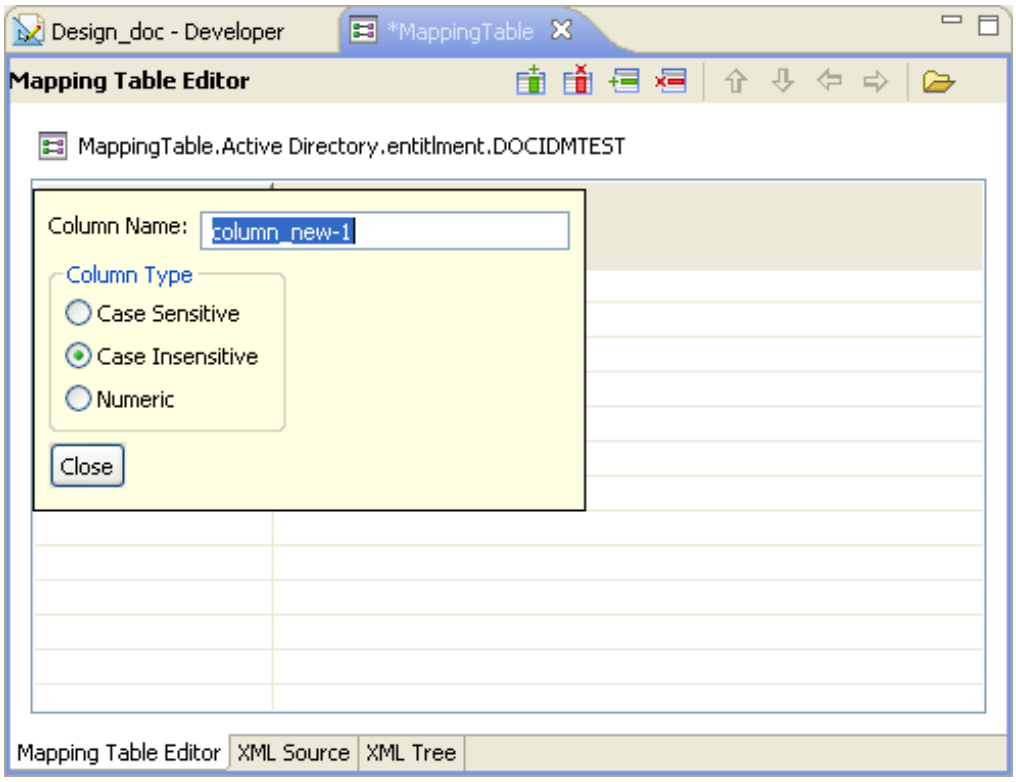
   Select **Open the editor after creating the object** to open the Mapping Table editor.



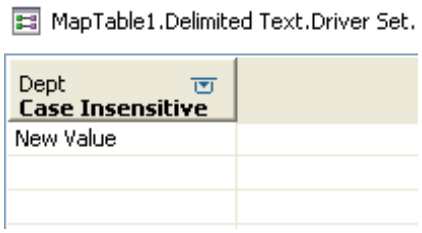
3. In the File Conflict message, click **Yes** to save the project before opening the Mapping Table editor.
4. In the Mapping Table editor, select **column_new-1**.

5. Specify a column name and data type, then click **Close**.

   Column names must be unique. The data type lets you specify if the column values are **Case Sensitive**, **Case Insensitive**, or **Numeric**.



6 Select **New Value** to specify a cell value.



7 (Optional) To add another column, click the **Add Column** icon , then repeat Step 4 and Step 5.

8 (Optional) To add another row, click the **Add Row** icon , then repeat Step 6.

9 Press Ctrl+S to save the mapping table object.

10 Continue with "Adding a Mapping Table Object to a Policy" on page 60.

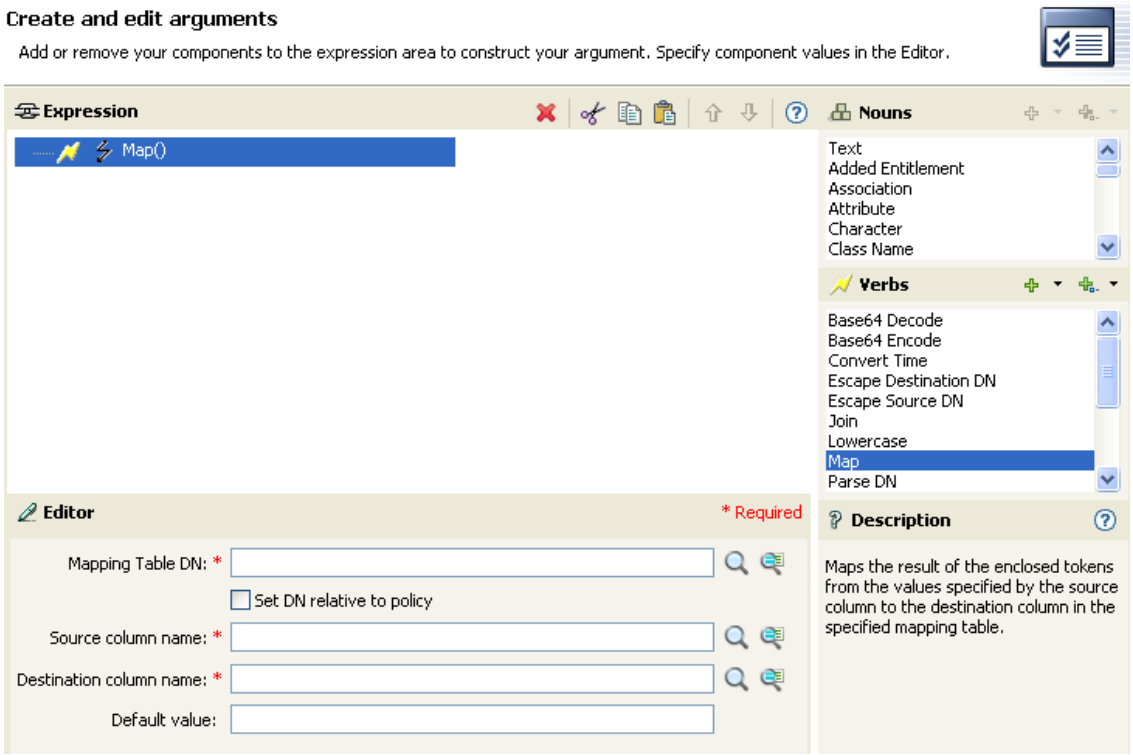
Adding a Mapping Table Object to a Policy

1 Either create a policy to use the mapping table in, or select an existing policy to edit.

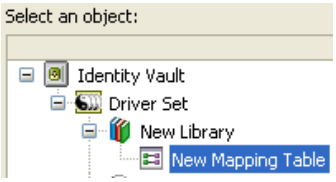
2 Launch the Argument Builder in the Policy Builder.

For information on how to access the Argument Builder, see "Launching the Argument Builder" on page 46.

3 Double-click **Map** from the list of verbs to add it to the expression panel.

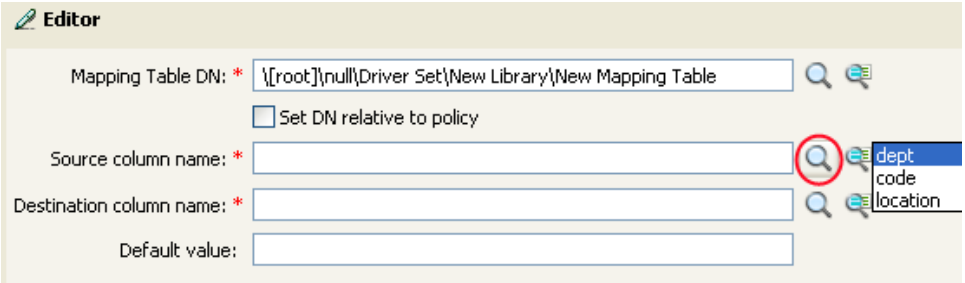


4 In the **Mapping Table DN** field, browse to and select the mapping table object created in "Creating a Mapping Table Object" on page 59, then click **OK**.

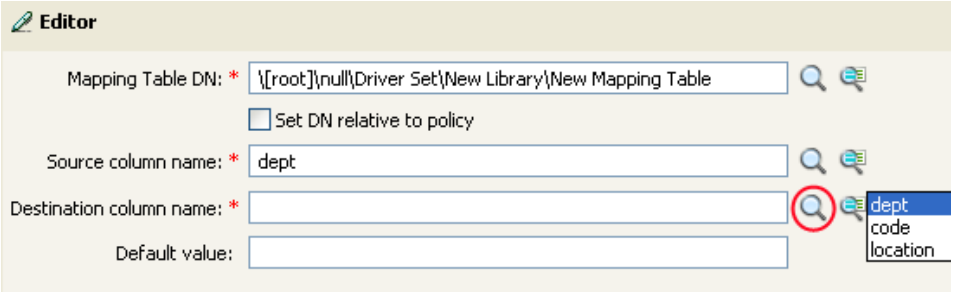


5 Select whether the mapping table DN is set relative to the policy or not.

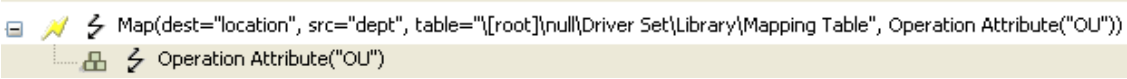
6 Select the source column name by clicking the **Browse** icon.



7 Select the destination column name by clicking the **Browse** icon.



The mapping table can be used in any manner at this point. In this example, the OU attribute is populated with the value derived from the mapping table.



Editing a Mapping Table Object

Designer provides the following options to edit the mapping table:

Table 3-5 *Editing Options for the Mapping Table Editor*

Option	Description
Undo Add Column	Undoes the last action performed in the table.
Redo Add Column	Redoes the action that was undone.
Add Column	Inserts a column to the mapping table.
Add Row	Inserts a row to the mapping table.
Delete Column	Deletes a column from the mapping table.
Delete Row	Deletes a row from the mapping table.
Move Row Up	Moves the selected row up in the mapping table.
Move Row Down	Moves the selected row down in the mapping table.
Move Column Left	Moves the selected column left in the mapping table.
Move Column Right	Moves the selected column right in the mapping table.

The Mapping Table Editor also supports keyboard shortcuts for several of its operations:

Table 3-6 Keyboard Shortcuts for the Mapping Table Editor

Keyboard Shortcut	Description
Ctrl+Shift+Insert	Insert a column to the right of the current column.
Ctrl+Shift+Delete	Delete the current column. You are prompted to confirm the deletion.
Ctrl+Shift+C	Rename the current column. Opens the Column Edit dialog box.
Alt+Insert	Insert a row below the current row.
Alt+Delete	Delete the current row. You are prompted to confirm the deletion.
Ctrl+Up Arrow	Navigate up one row.
Ctrl+Down Arrow	Navigate down one row.
Ctrl+Left Arrow	Navigate left one column.
Ctrl+Right Arrow	Navigate right one column.

Importing Data from a CSV File

The Mapping Table editor allows you to import data that is stored in a CSV file. It then populates the table with the information in the CSV file. To import a CSV:

1. In an empty Mapping Table, select **Import From CSV file** .
2. Browse to and select the CSV file, then click **Open**.
3. Click **Yes** to overwrite your existing data.
4. Press Ctrl+S to save the data in the table.

Exporting Data to a CSV File

The Mapping Table editor allows you to export data to a CSV file. To export data to a CSV file:

1. When the data in the Mapping Table is ready to export, select **Export To CSV File** .
2. Click **Yes** to save this editor's changes and continue.
3. Specify a name and location for the CSV file, then click **Save**.

Testing a Mapping Table Object

You can use the Policy Simulator to test the functionality of the mapping table. The Policy Simulator tests the mapping table by testing the policy that is using the mapping table. For more information, see Chapter 8, "Testing Policies with the Policy Simulator," on page 143.

Global Configuration Value Definition Editor

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

To create a Global Configuration object:

1. Right-click an Identity Vault, driver set, or driver in the Modeler, then click **New** > **Global Configuration.**
2. Specify a name for the Global Configuration object, then click **OK**.
3. Double-click the new object in the Outline view, then click **GCVs**.
4. Click **Add**, then fill in the following fields to create the GCVs:

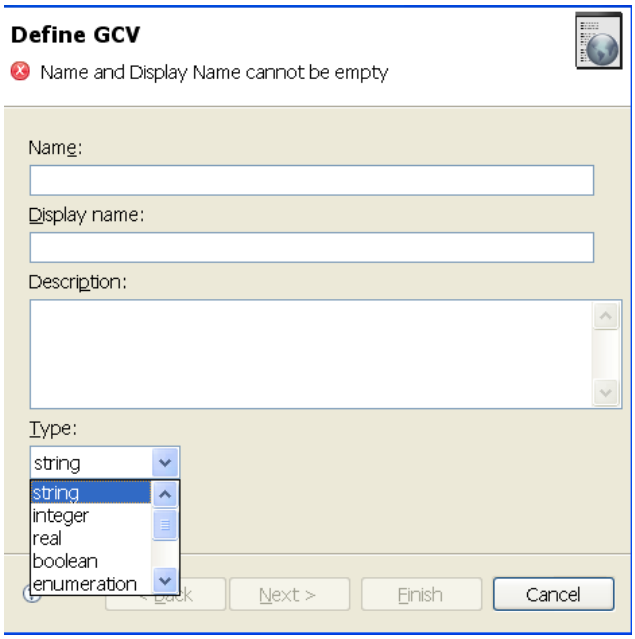
   Name: Specify the name for the GCV.

   Display Name: Specify the name that is displayed for the GCV.

   Description: Specify a description of the GCV.

   Type: Select the type of GCV that you are creating and fill in the appropriate information for the type. The options are:

   - **string:** Select whether the string is contained on multiple lines by clicking **Multi-line.** A string value is a sequence of Unicode characters.
   - **integer:** Specify the minimum value and maximum value of the integer. An integer value contains one or more Unicode characters.
   - **real:** Specify the minimum value and maximum value of the real. A real value describes a real or floating point number.
   - **boolean:** The boolean value is either true or false.
   - **enumeration:** The enumeration value is a defined set of strings. To create an enumeration value, click **Add**, then specify the display name and value for the enumeration, then click **OK**. Repeat this process for each string you add to the enumeration value.
   - **dn:** The dn value is obtained from the DN syntax of an object in the Identity Vault. You must select the dn syntax type and the namespace of the DN.
     - **Syntax:** The syntax options are LDAP, slash, qualified-slash, dot, qualified-dot, and custom.
     - **DN Space:** Select whether the namespace is from the connected system (application) or from OpenText Identity Manager (dirxml).
   - **list:** The list value contains an ordered list of zero or more strings. Specify the delimiter character that is used to separate the items in the list.
   - **password-ref:** Specify the key value of a named password. A named password key valued can be any non-empty sequence of Unicode characters.
   - **structured:** A structured value is similar to structures in the C programming language. A structure contains to fundamental parts: a template that defines a set of simpley types, and zero or more instances that contain the actual values of the structured control value. Specify the minimum and maximum instances, as well as the value separator and instance separator for the structure value.



5 Click **Finish**.

6 Repeat Step 4 and Step 5 for each GCV you want to add, then click **OK** to save the changes.

For more information about defining GCVs, you can refer to Explaining GCVs (https://www.netiq.com/communities/cool-solutions//explaining-gcvs-part-1).

Namespace Editor

The Policy Builder enables you to use multiple XML namespaces within your XML documents. You launch the Namespace editor when you access the following DirXML Script elements in the Policy Builder:

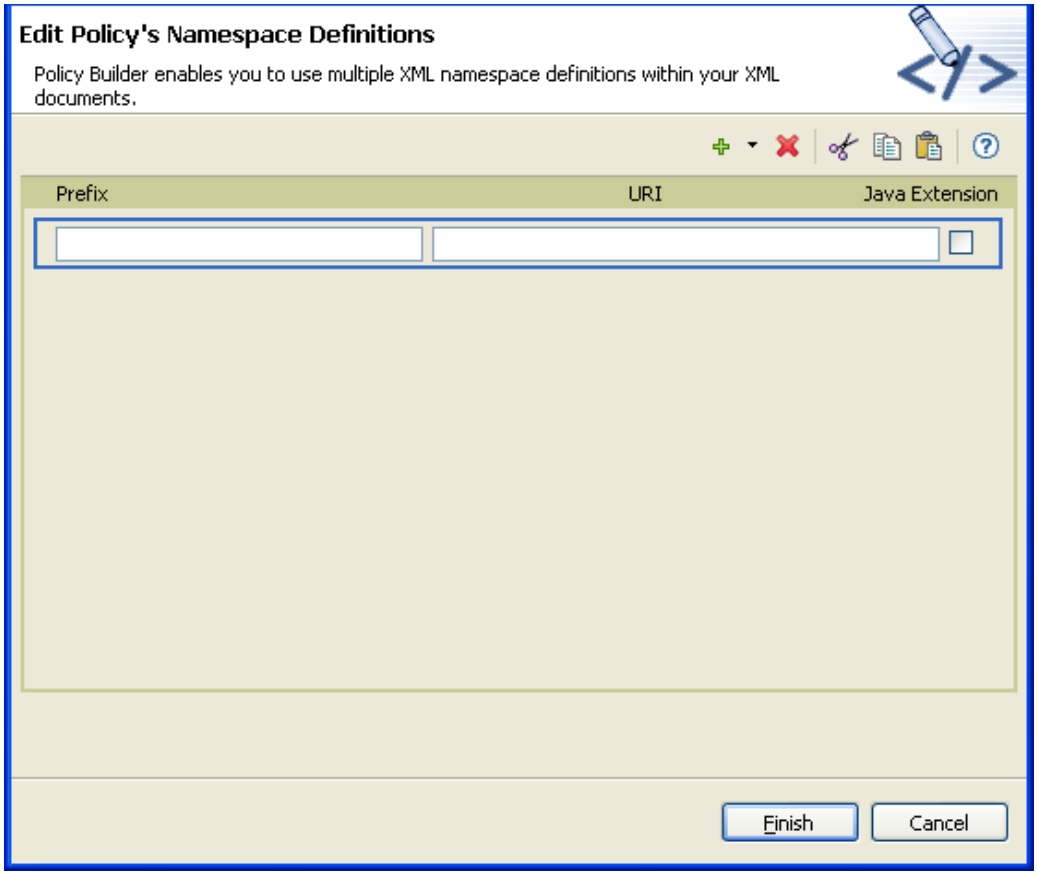
- Append XML Element (page 249)
- Append XML Text (page 251)
- Clone By XPath Expressions (page 258)
- Set XML Attribute (page 329)
- Strip XPath Expression (page 334)
- XPath (page 382)

1 Click the **Edit the policy's namespace definitions** icon.

2 Specify the namespace prefix.

3 Specify the URI.

4 Do not select **Java Extension**.



You can also access Java* classes through XPath by using XML namespaces. To create a namespace for a Java class, specify the namespace prefix in the **Name** field, the class name in the **URI** field, and select the **Java Extension** check box.

Accessing Java Classes Using Namespaces

OpenText provides several OpenText Identity Manager Java classes that can be called by using XPath expressions from the Policy Builder. For more information, visit https://www.netiq.com/documentation/identity-manager-developer/driver-developer-kit.html.

The Java Developer Kit (JDK) also provides several useful classes, such as java.lang.String, and java.lang.System. References for these classes are available with the JDK.

For additional information on using XPath and the OpenText Java classes listed above, consult the DirXML Driver Developer Kit (https://www.netiq.com/documentation/identity-manager-developer/driver-developer-kit/dirxmlfaq.html).

Local Variable Selector

Policies use local variables and they have different scopes. A local variable is defined for a specific policy or it is defined for a driver. If a local variable scope is set to driver, then any policy in the driver can use this variable.

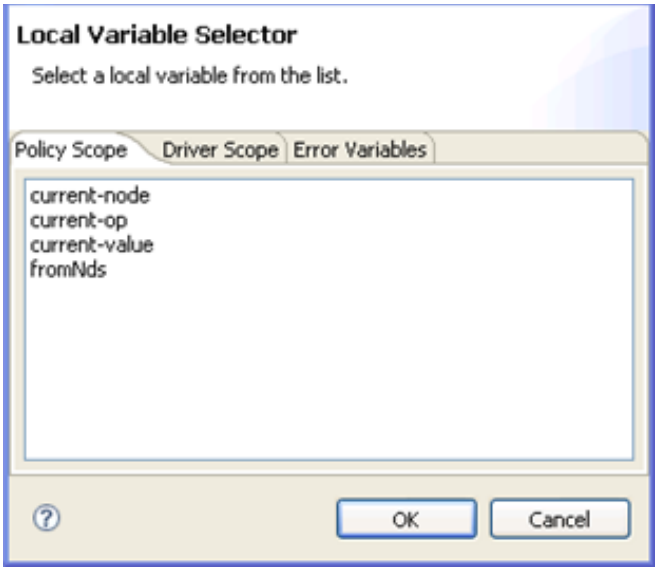
The Policy Builder contains a Local Variable Selector that allows you to select any local variables that have been defined for use in the selected policy.

The Local Variable Selector is accessed through the following actions, conditions, and tokens:

- If Local Variable (page 208)
- Set Local Variable (page 315)
- Local Variable (page 361)

The Local Variable Selector displays the following tabs:

Figure 3-10 *Local Variable Selector*



Policy Scope: Lists any local variables with a scope of policy.

Driver Scope: Lists any local variables with a scope of driver.

Error Variables: Lists local variables that are set if an error is encountered during the execution of the policy that contains the following actions:

- Add Resource (page 243)
- Add Role (page 245)
- Clear SSO Credential (page 257)
- Find Matching Object (page 270)
- Remove Resource (page 298)
- Remove Role (page 296)
- Send Email (page 304)

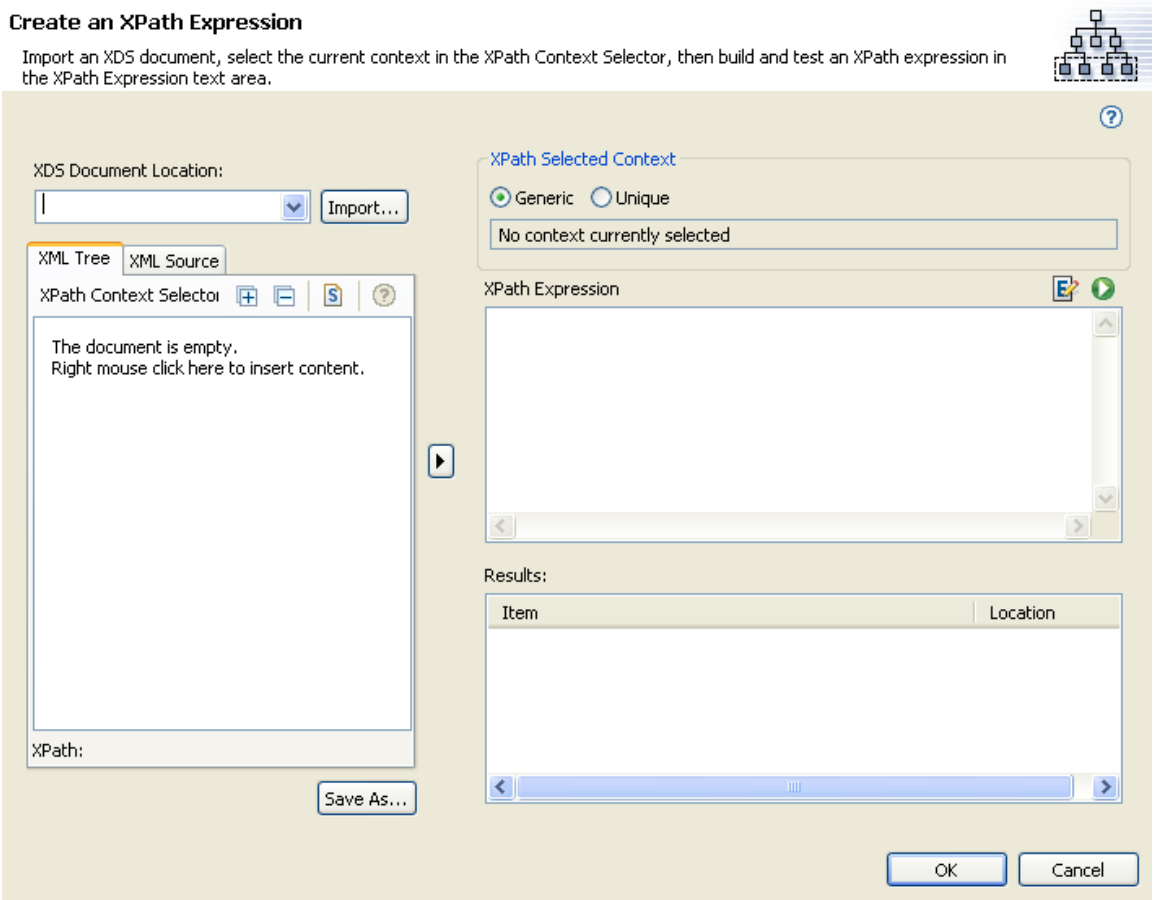
- Send Email from Template (page 307)
- Set SSO Credential (page 327)
- Set SSO Passphrase (page 328)
- Start Workflow (page 330)

OpenText Identity Manager allows you to view the content of the error variables and take appropriate actions.

4 Using the XPath Builder

The XPath Builder is a powerful tool that allows you to build and test an XPath expression against any XML document. You can test different expressions against an XDS document and modify the XDS document while testing the expression. For more information about XPath expression, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Figure 4-1 *XPath Builder*



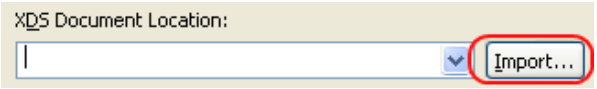
To use the XPath Builder:

1 In the Policy Builder, select any of the following conditions or actions, then click the **Launch XPath Builder** icon .

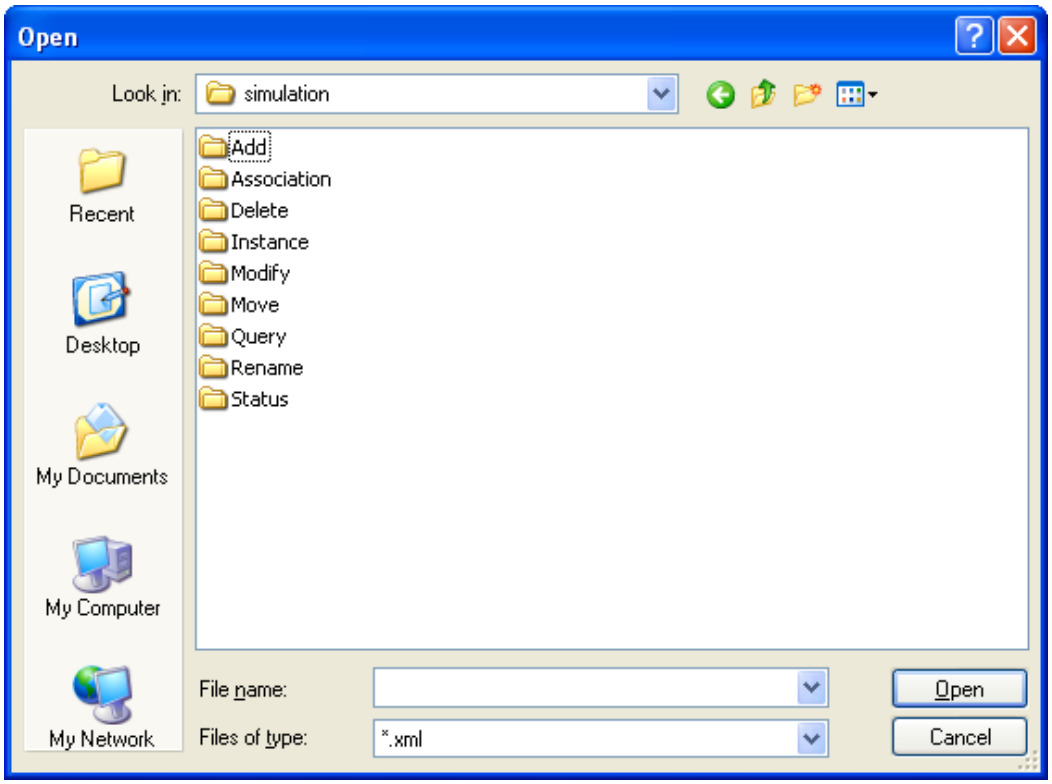
- If XPath Expression (page 232)
- Append XML Element (page 249)
- Append XML Text (page 251)
- Clone By XPath Expressions (page 258)

- Set XML Attribute (page 329)
- Strip XPath Expression (page 334)

2 Select **Import** to browse to and select the XDS document to test.



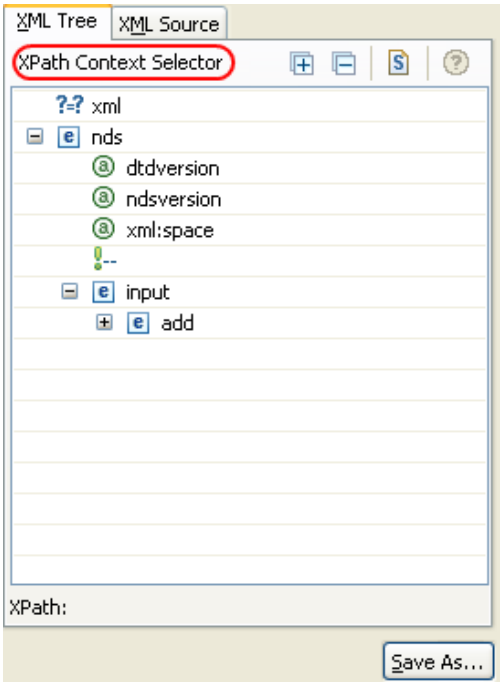
Designer comes with sample event files you can use to test the XPath expression against. The files are located in the plug-in `com.novell.designer.idm.policy_`*`version`*`\simulation`, where *version* is the current version of Designer. The events are Add, Association, Delete, Instance, Modify, Move, Query, Rename, and Status.



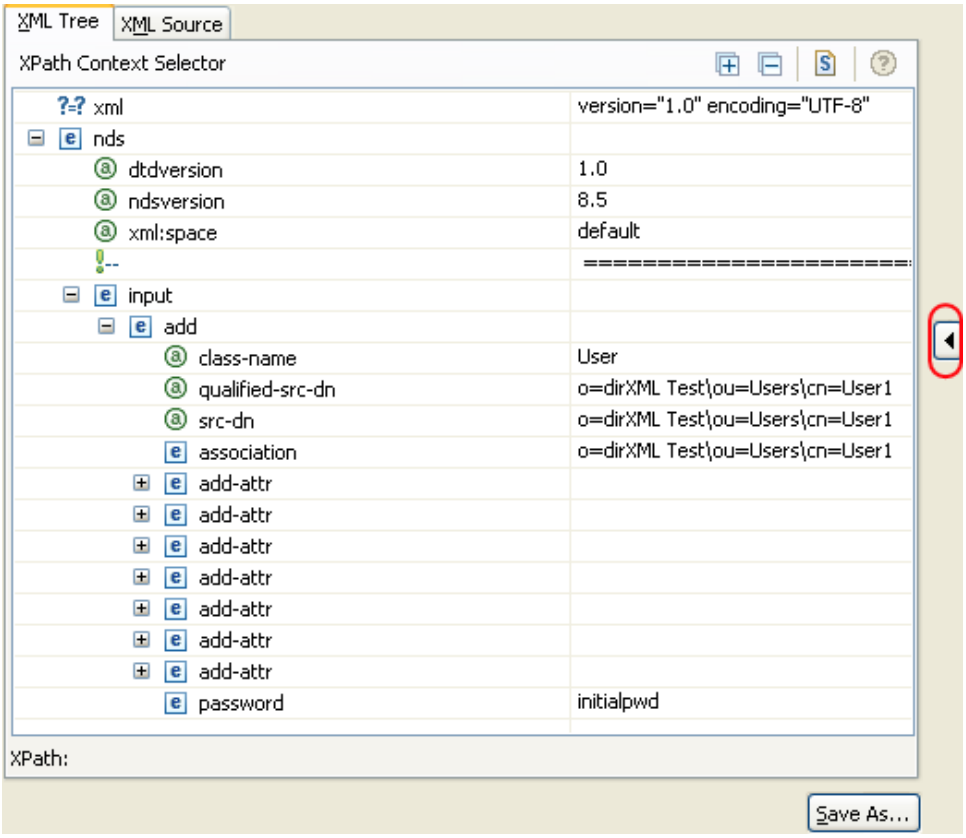
3 Double-click the folder to display the available events. Each event has different files you can select. For example, if you select **Add** you have three options: `Organization.xml`, `OrganizationalUnit.xml`, and `User.xml`. The file indicates the event. If you select `User.xml`, it is an Add event for a User object.

4 Select a file, then click **Open**.

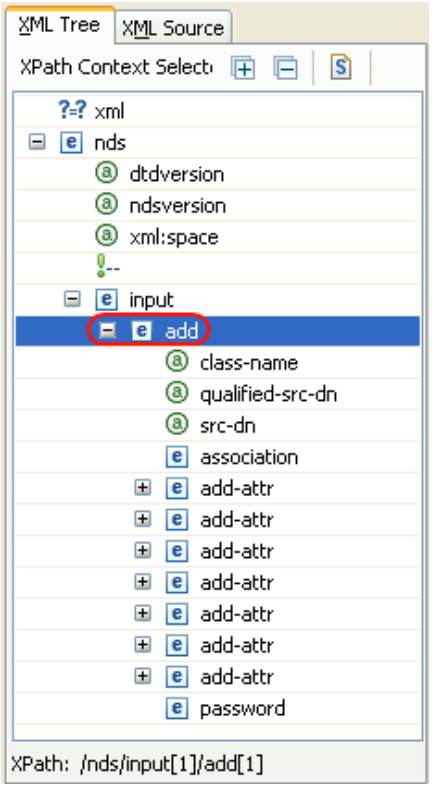
The input document is now displayed in the **XPath Context Selector** view. The **XML Source** tab allows you to use an XML source editor to edit the imported document, or an XML document from another editor can be copied and pasted into the source view. If you change the document, click **Save As** to save the changed document.



If you want to see the XDS document without scrolling, click the **Hide XPath Details** icon . To see the **XPath Expression** and **Results** windows, click **Show XPath Details** icon.



5 Select the current position in the document from which you want to start building your XPath expression.



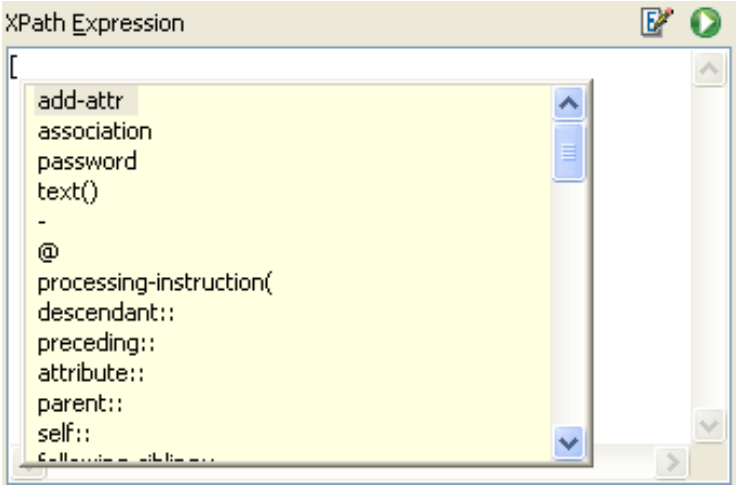
The XPath context that you have selected is displayed in the **XPath Selected Context** as shown.

6 Select **Generic** or **Unique**.

Generic searches the entire XML document to match the specified XPath expression. It returns results for each instance of the XPath expression. In this example, the XPath expression is “/nds/input/add”. It searches the entire XML document for each instance of add.

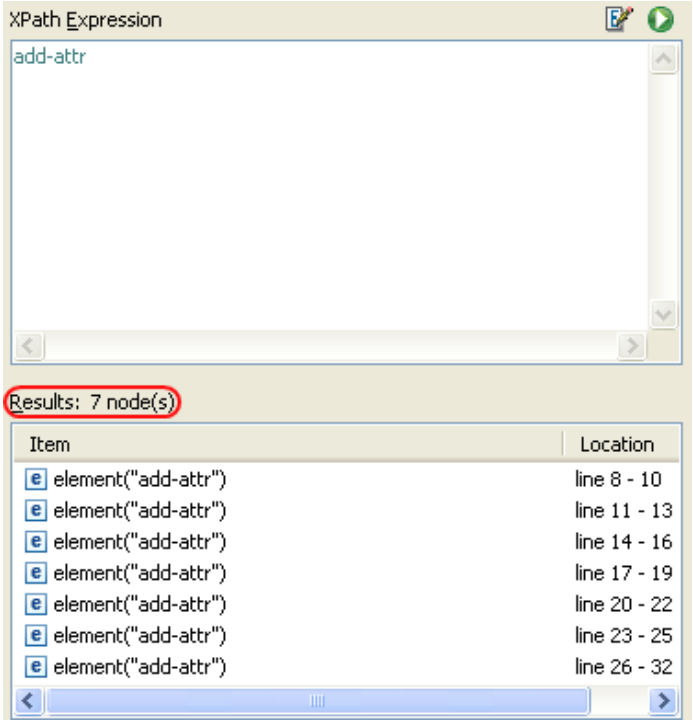
Unique searches the XML document until it finds a match and then stops. The unique XPath expression is “/nds/input[1]/add[1]”. It searches for the first instance of add and then stops. You can specify which instance you want to use by selecting the next instance of the XPath element in the **XML Context Selector**.

7 Specify an XPath expression in the **XPath Expression** field.



NOTE: Using the keystroke combination Ctrl+Space+3, /, [, or (triggers code completion. The expression is evaluated up until the cursor location, and insertable elements are shown in a drop-down box.

The results of your XPath expression appear in the **Results** text area below.



If the XPath editor does not evaluate the expression, click the **Evaluate XPath expression** icon to force the XPath Builder to evaluate the expression.

8 (Optional) Click the **ECMA Expression Editor** icon to use a valid ECMAScript expression instead of an XPath expression.

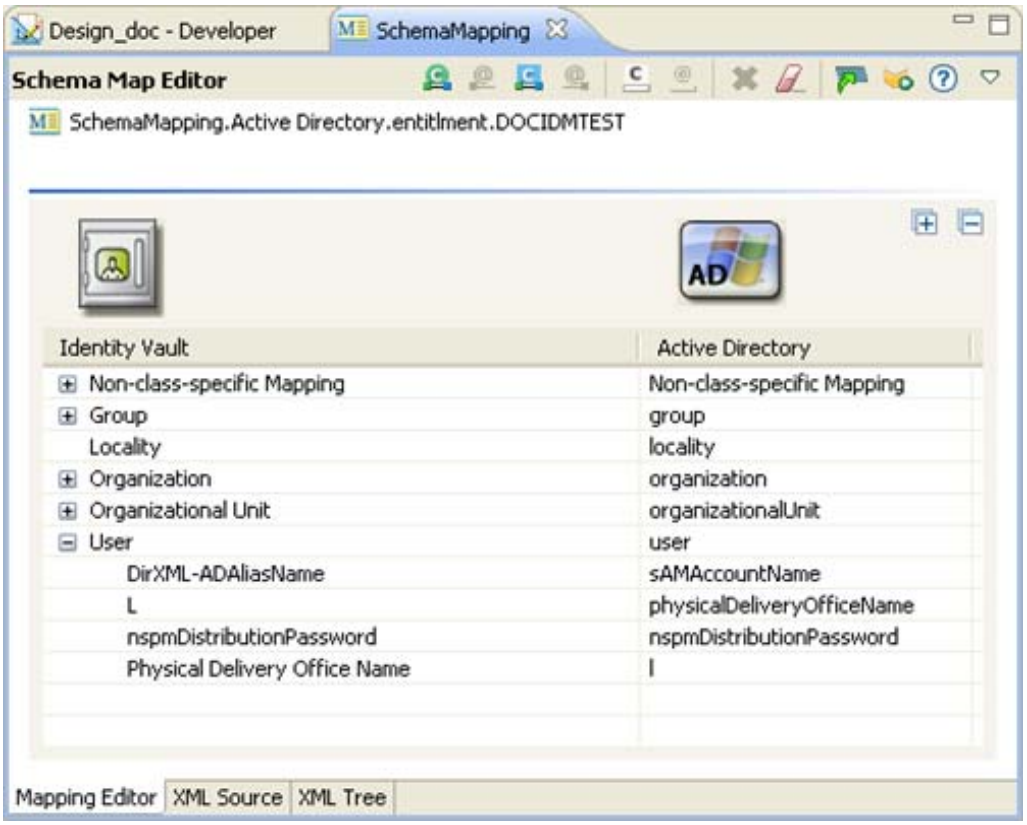
9 When you are finished building and testing an XPath expression, click **OK** to close the XPath Builder.

The text displayed in the **XPath Expression** is placed into the policy that you are editing.

5 Defining Schema Map Policies

Schema Map policies map class names and attribute names between the Identity Vault namespace and the application namespace. All documents passed between the OpenText Identity Manager engine and the application shim in either direction on either channel are passed through the Schema Map policy. There is one Schema Map policy per driver.

Figure 5-1 *The Schema Map Editor*



NOTE: The Schema Map editor is for creating and managing schema map policies. If you want to manage the actual schema on the Identity Vault or Application, use the Manage Schema tool, which is accessible by clicking the pull-down menu , then selecting **Manage Identity Vault Schema** or **Manage Application Schema** .

For more information, see "Managing the Schema" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

This section includes the following topics:

- "Using the Schema Map Editor" on page 76
- "Editing a Schema Map Policy" on page 79

- "Testing Schema Map Policies" on page 86
- "Exporting and Importing with the Schema Map Editor" on page 86
- "Accessing the Schema Map Policy in XML" on page 86
- "Additional Schema Map Policy Options" on page 87

Using the Schema Map Editor

The Schema Map editor allows you to edit the Schema Map policies. This section includes the following topics:

- "Accessing the Schema Map Editor" on page 76
- "Navigating the Schema Map Editor" on page 77
- "Understanding the Schema Map Editor Toolbar" on page 78

Accessing the Schema Map Editor

There are three different ways to access the Schema Map editor in Designer:

Outline View To open the Schema Map editor from the Outline view:

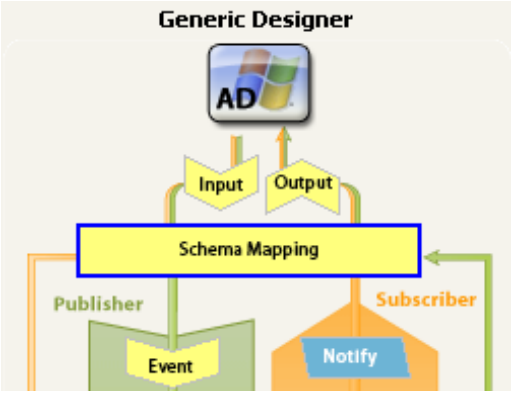
1. In an open project, click the **Outline** tab.
2. Click **Show Model Outline** .
3. Expand the driver where you want to manage the schema map policy.
4. Double-click the **Schema Map** icon to launch the Schema Map editor.

   You can also right-click the icon, then select **Edit**.

Policy Flow View To open the Schema Map editor from the Policy Flow view:

1. In an open project, click the **Outline** tab.
2. Click **Show Policy Flow** .
3. Double-click the Schema Mapping object, select the **Schema Mapping** policy, then click **Edit** to launch the Schema Map Editor.

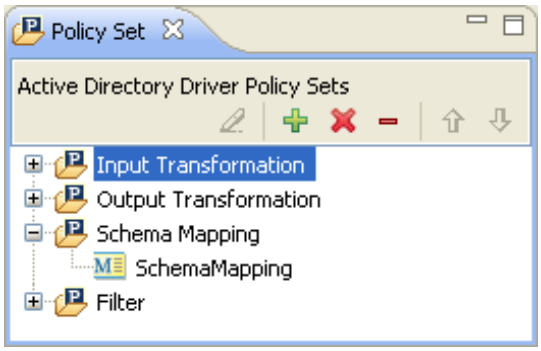
   You can also right-click the **Schema Mapping** object, then select **Edit Policy** to launch the Schema Map Editor.



Policy Set View To open the Schema Map editor from the Policy Set view:

1. In an open project, click the **Outline** tab.
2. Click the **Show Model Outline** icon.
3. In the Outline view, select the appropriate driver object.
4. In the Policy Set view, open the **Schema Mapping** folder, then double-click the Schema Mapping policy to launch the Schema Map editor.

   You can also right-click the Schema Mapping policy, then click **Edit** to launch the Schema Map editor.



Navigating the Schema Map Editor

The Schema Map Editor uses standard point-and-click navigation. However, it also provides keyboard-based navigation options as described in Table 6-1.

NOTE: The Schema Map Editor lets you order the list of mapped classes and attributes alphabetically (ascending or descending.) To do so, click either the gray Identity Vault header or the gray application datastore header that appears above the list of mapped classes. If you first select a class mapping and then click one of the headers, only the attributes within the class mapping are ordered.

Table 5-1 Schema Map Editor Keyboard Support

Keystroke	Description
Up-arrow	Moves the cursor up in the Schema Map editor.
Down-arrow	Moves the cursor down in the Schema Map editor.
Left-arrow	Collapses the information displayed
Right-arrow	Expands the information displayed.
Insert	Adds a class.
Ctrl+Insert	Adds an attribute.
Delete	Deletes the selected items.
Enter	Opens edit mode for the currently selected field. Press Enter a second time to commit the change in Schema Map editor.
Esc	Exits the edit mode.

Understanding the Schema Map Editor Toolbar

The Schema Map editor includes a toolbar that provides access to the following features. Several of these features, along with an option to **Edit** a selected mapping, is also available from a drop-down menu by right-clicking in the Schema Map editor.

Tool	Description
	Insert Identity Vault Class launches a dialog box from which you can add a new ID Vault class, and its associated attributes, to the schema map. For more information, see "Adding an Identity Vault Class or Attribute" on page 80.
	Insert Identity Vault Attribute launches a dialog box from which you can add additional attributes to an existing ID Vault class in the schema map. For more information, see "Adding an Identity Vault Class or Attribute" on page 80.
	Insert Application Class launches a dialog box from which you can add a new Application class, and its associated attributes, to the schema map. For more information, see "Adding an Application Class or Attribute" on page 82.
	Insert Application Attribute launches a dialog box from which you can add additional attributes to an existing Application class in the schema map. For more information, see "Adding an Application Class or Attribute" on page 82.
	Insert Class Row adds an empty class row to the schema map. You can then populate the class fields manually or by selecting from the drop-down menu of available classes.
	Insert Attribute Row adds an empty attribute row to the selected class in the schema map. You can then populate the attribute fields manually or by selecting from the drop-down menu of available attributes.
	Delete deletes the selected class or attribute mappings from the schema map.

Tool	Description
	Clear All Items deletes all class and attribute entries from the schema map.
	Synchronize with the Filter Editor instructs the Schema Map editor to update the Filter policy with any schema mappings you have added in the Schema Map editor. The Schema Map editor does not synchronize deleted entries to the Filter policy. For more information about filter policies and the Filter editor, see Chapter 6, "Controlling the Flow of Objects with the Filter," on page 91.
	Launch Policy Simulator launches the Policy Simulator. For more information, see Chapter 8, "Testing Policies with the Policy Simulator," on page 143.
	Help launches the context-sensitive help for the Schema Map editor.
	The pull-down menu opens a secondary menu of schema map editor tools, including the following: **Save to File** exports the current schema map to an XML file. **Import from File** imports a schema map from a previously saved XML file. **Manage Identity Vault Schema** launches the Manage Schema tool. For more information, see "Managing the Schema" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* . **Manage Application Schema** launches the Manage Schema tool. For more information, see "Managing the Schema" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* . **Refresh Application Schema** queries a live application for its current schema. This lets you update the application schema in Designer as it changes on the live system.

Editing a Schema Map Policy

The Schema Map editor allows you to create and edit schema map policies. This section includes the following topics:

- "Adding or Deleting Classes and Attributes" on page 80
- "Refreshing the Application Schema" on page 84
- "Editing Items" on page 84
- "Sorting Schema Map Entries" on page 85
- "Managing the Schema" on page 85

For information about exporting and importing a schema map policy, see "Exporting and Importing with the Schema Map Editor" on page 86.

Adding or Deleting Classes and Attributes

There are three types of classes or attributes you can add to a schema map. The process for adding each type of class or attribute varies.

When you add or remove a class or attribute in the Schema Map policy, Designer updates relevant filters at the same time. For more information about filters, see Chapter 6, "Controlling the Flow of Objects with the Filter," on page 91.

- "Adding an Identity Vault Class or Attribute" on page 80
- "Adding an Application Class or Attribute" on page 82
- "Adding a Non-class-specific Attribute Mapping" on page 83
- "Deleting a Class or Attribute Mapping" on page 83

Adding an Identity Vault Class or Attribute

You can both add new Identity Vault classes and attributes to a schema map, and add additional Identity Vault attributes to an existing class mapping.

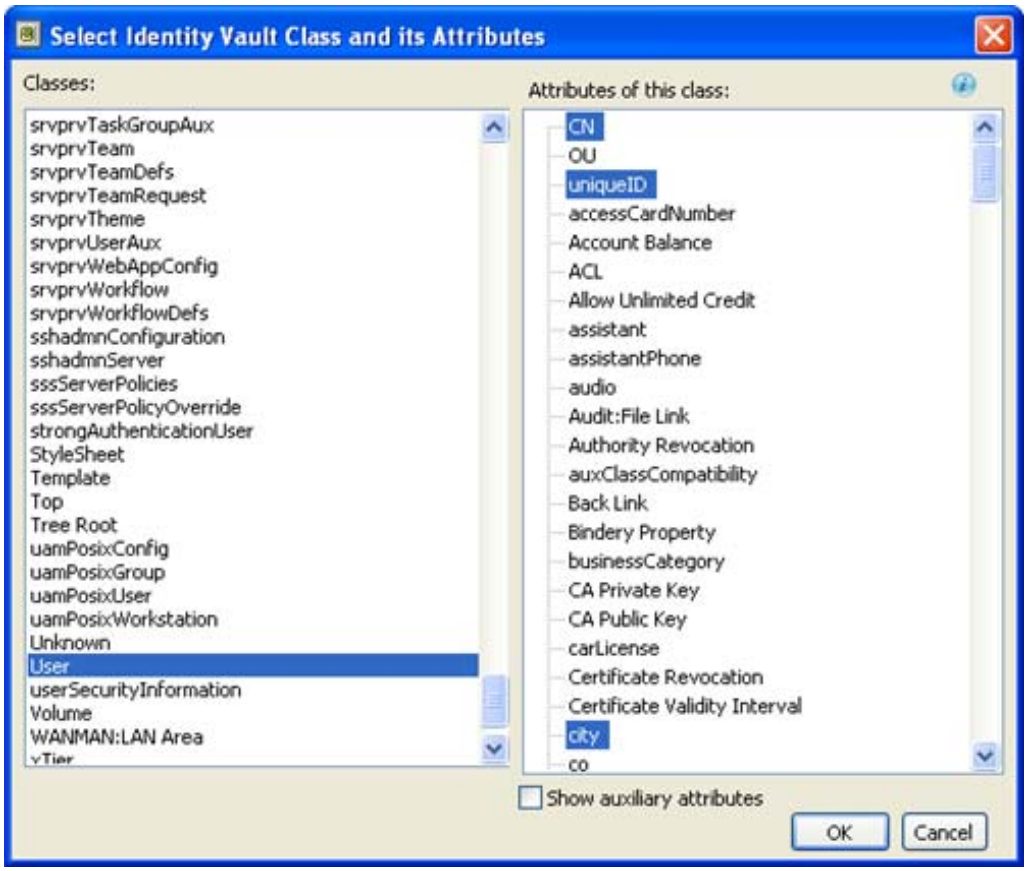
To add a new Identity Vault class and attributes to a schema map:

1. In the Schema Map Editor, select **Insert Identity Vault Class** .

   You can also right-click in the Schema Map editor, then click **Insert Identity Vault Class**.

2. In the Select Identity Vault Class and its Attributes page, select a class and the relevant class attributes to add to the schema map, then click **OK**.

   Use Shift+click and Ctrl+click to select multiple attributes, if desired.



3 In the Schema Map Editor, double-click each class and attribute you added to the schema map, then specify the appropriate Application class (or attribute) to which you want to map it.

You can either select the class or attribute name from the drop-down list, or type it in the field manually.

Identity Vault	Active Directory
⊞ Non-class-specific Mapping	Non-class-specific Mapping
⊟ Group	group
DirXML-ADAliasName	sAMAccountName
GID	guid
Locality	locality
⊞ Organization	organization
⊞ Organizational Unit	organizationalUnit
⊞ User	user

4 To save the schema map changes, select **File** > **Save**.

To add additional Identity Vault attributes to an existing class mapping:

1 In the Schema Map Editor, select a class mapping, then select **Add Identity Vault Attributes**.

You can also right-click in the Schema Map editor, then select **Insert Identity Vault Attributes**.

2 In the Select ID Vault Attributes page, select the desired attributes to add to the class mapping, then click **OK**.

Use Shift+click and Ctrl+click to select multiple attributes, if desired.

3 In the Schema Map Editor, double-click each attribute you added to the schema map, then specify the appropriate Application attribute to which you want to map it.

You can either select the attribute from the drop-down list, or type it in the field manually.

Identity Vault	Active Directory
⊞ Non-class-specific Mapping	Non-class-specific Mapping
⊟ Group	group
DirXML-ADAliasName	sAMAccountName
GID	guid
Locality	locality
⊞ Organization	organization
⊞ Organizational Unit	organizationalUnit
⊞ User	user

4 To save the schema map changes, select **File** > **Save**.

Adding an Application Class or Attribute

You can both add new Application classes and attributes to a schema map, and add additional Application attributes to an existing class mapping.

IMPORTANT: To view an application's schema classes and attributes, the driver must be able to retrieve the schema information from a live application environment. This occurs automatically when a driver starts (right-click the driver, then select **Live** > **Start Driver**). However, you can refresh the application schema at any time by selecting **Refresh Application Schema**.

To add a new Attribute class and attributes to a schema map:

1 In the Schema Map Editor, select **Insert Application Class**.

You can also right-click in the Schema Map editor, then click **Insert Application Class**.

2 In the Select Application Class and its Attributes page, select a class and the relevant class attributes to add to the schema map, then click **OK**.

Use Shift+click and Ctrl+click to select multiple attributes, if desired.

3 In the Schema Map Editor, double-click each class and attribute you added to the schema map, then specify the appropriate Application class (or attribute) to which you want to map it.

You can either select the class or attribute name from the drop-down list, or type it in the field manually.

Identity Vault	Active Directory
⊞ Non-class-specific Mapping	Non-class-specific Mapping
⊟ Group	group
DirXML-ADAliasName	sAMAccountName
GID	guid
Locality	locality
⊞ Organization	organization
⊞ Organizational Unit	organizationalUnit
⊞ User	user

4 To save the schema map changes, select **File** > **Save**.

To add additional Application attributes to an existing class mapping:

1 In the Schema Map Editor, select a class mapping, then select **Insert Application Attributes**.

   You can also right-click in the Schema Map editor, then select **Insert Identity Vault Attributes**.

2 In the Select App Attributes page, select the desired attributes to add to the class mapping, then click **OK**.

   Use Shift+click and Ctrl+click to select multiple attributes, if desired.

3 In the Schema Map Editor, double-click each attribute you added to the schema map, then specify the appropriate Identity Vault attribute to which you want to map it.

   You can either select the attribute from the drop-down list, or type it in the field manually.

Identity Vault	Active Directory
⊞ Non-class-specific Mapping	Non-class-specific Mapping
⊟ Group	group
DirXML-ADAliasName	sAMAccountName
GID	guid
Locality	locality
⊞ Organization	organization
⊞ Organizational Unit	organizationalUnit
⊞ User	user

4 To save the schema map changes, select **File** > **Save**.

Adding a Non-class-specific Attribute Mapping

Sometimes an attribute mapping doesn't apply to a specific class. In this case you can define the attribute mapping in the Non-class-specific container.

To add a non-class-specific attribute mapping:

1 Select the **Non-class-specific Mapping** entry in the Schema Map Editor.

2 Add the appropriate attribute mapping using one of the methods described previously.

   For more information, see "Adding an Identity Vault Class or Attribute" on page 80 and "Adding an Application Class or Attribute" on page 82.

Deleting a Class or Attribute Mapping

If you do not want an Identity Vault class or an attribute to be mapped to an Application class or attribute, the best practice is to completely remove the class or the attribute from the Schema Map policy. To remove multiple classes or attributes at the same time, use Ctrl-click or Shift-click to select more than one class or attribute at a time.

Identity Vault	Active Directory
Non-class-specific Mapping	Non-class-specific Mapping
Group	group
Locality	locality
Organization	organization
L	physicalDeliveryOfficeName
Physical Delivery Office Name	l
Organizational Unit	organizationalUnit
User	user
DirXML-ADAliasName	sAMAccountName
L	physicalDeliveryOfficeName
nspmDistributionPassword	nspmDistributionPassword
Physical Delivery Office Name	l

You can add or remove attributes and classes from the Schema Map policy in the following ways:

- Select the classes or attributes you want to remove, then right-click and select **Delete**.
- Select the classes or attributes you want to remove, then click **Delete** in the Schema Map editor toolbar.
- Select the classes or attributes you want to remove, then press the Delete key.

You can also delete all classes and attributes at once by selecting **Clear All Items** .

Refreshing the Application Schema

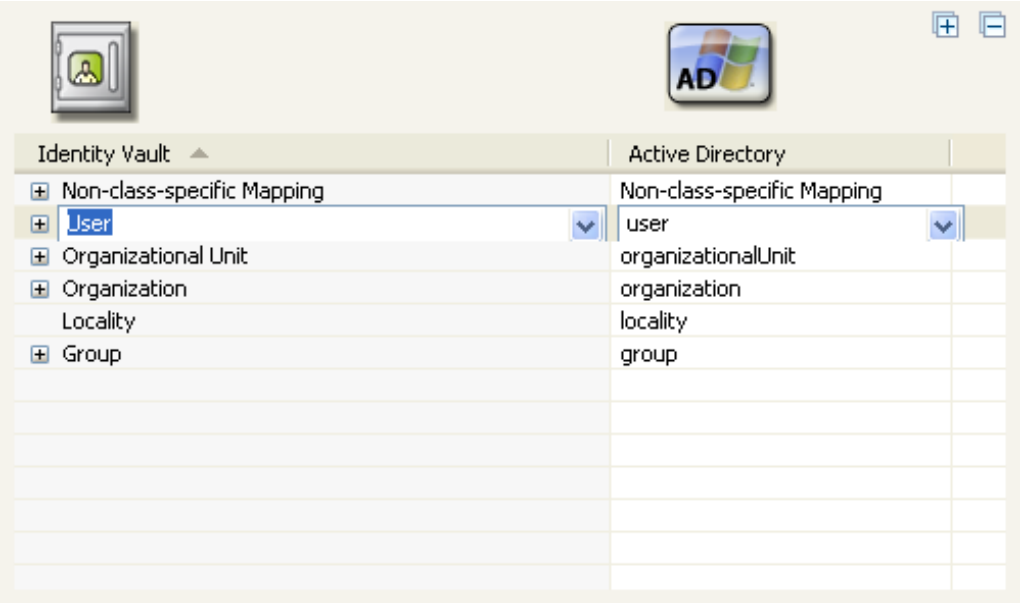
If you have modified the schema in the application, these changes need to be reflected in the Schema Map policy. To make the new schema available, click the toolbar pull-down menu, then select **Refresh Application Schema** .

Refreshing the application schema requires a connection to the live application because the application driver must be able to query the application for the updated schema.

Editing Items

To edit a mapping, double-click the selected row. An in-place editor appears, allowing you to edit the mapping.

Figure 5-2 *In-line Edits in the Schema Map Editor*



Sorting Schema Map Entries

The Schema Map editor allows you to sort entries in ascending/descending order by clicking on the column heading. Click the Identity Vault heading to sort entries based on Identity Vault items. Click the connected system heading to sort entries based on connected system items.

Managing the Schema

Designer allows you to manage the Identity Vault schema and any connected system's schema. You can import the schema, modify it, and deploy the changed schema back into the Identity Vault or the Application.

To manage the Identity Vault schema, click the pull-down menu , then select **Manage Identity Vault Schema** . This opens the Manage Schema tool and displays information about the classes and attributes in the Identity Vault schema.

To manage the Application schema, click the pull-down menu , then select **Manage Application Schema** . This opens the Manage Schema tool and displays information about the classes and attributes in the Application schema.

For more information about how to manage the schema, see “Managing the Schema” in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Testing Schema Map Policies

Designer comes with a tool called the Policy Simulator. It allows you to test your policies without implementing them in a production environment. You can launch the Policy Simulator through the Schema Map editor to test your policy after you have modified it.

To access the Policy Simulator and test the Schema Map policy:

1. Click the **Launch Policy Simulator** icon in the toolbar.
2. Select **To Identity Vault** or **From Identity Vault** as the simulation point of the Schema Map policy.

For more information on the Policy Simulator, see Chapter 8, "Testing Policies with the Policy Simulator," on page 143.

Exporting and Importing with the Schema Map Editor

Designer allows you to export a schema map policy document to an XML file. It also allows you to Import an XML file from a particular point on the file system to the Schema Map Editor.

Exporting a Schema Map Policy

Schema Map policies can be exported from the editor and saved as an XML file located in the file system.

1. In the Schema Map editor, click the pull-down menu , then select **Save to File** .
2. Specify a filename and location where you want to export your schema map policy, then click **Save**.

Importing a Schema Map Policy

The Exported policies which were saved as XML files on the file system can be re-imported to the Schema Map editor. This functionality saves you the effort of redoing the class or attribute mappings again. To import a schema map policy:

1. In the Schema Map editor, click the pull-down menu , then select **Import from File** .
2. In the Import a Schema Map File dialog box, browse to the schema file you want to import, then click *Open*.

   Specify whether you want to append the imported schema mappings to the existing schema map, or replace the existing schema map with the imported schema map.

Accessing the Schema Map Policy in XML

Designer enables you to view, edit, and validate the XML by using an XML editor. Click the **XML Source** tab or the **XML Tree** tab to access the XML editor. For more information about the XML editor, see "The NetIQ XML Editor" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Additional Schema Map Policy Options

When you right-click a Schema Map policy, there are multiple options presented in the Outline view, the Policy Flow view, and the Policy Set view.

- "Outline View Additional Options" on page 87
- "Policy Flow View Additional Options" on page 88
- "Policy Set View Additional Options" on page 89

Outline View Additional Options

There are additional options to manage the Schema Map policy in the Outline view. Right-click the Schema Map policy in the Outline view to see the additional options.

Table 5-2 *Schema Map Policy Options in the Outline View*

Option	Description
Edit	Launches the Schema Map editor. For more information, see "Editing a Schema Map Policy" on page 79.
Copy	Creates a copy of the Schema Map policy.
Save As	Saves the Schema Map policy as a `.xml` file.
Simulate	Tests the Schema Map policy. For more information, see "Testing Schema Map Policies" on page 86.
Export to Configuration File	Saves the Schema Map policy as a `.xml` file.
Live > Deploy	Deploys the Schema Map policy into the Identity Vault. For more information, see "Deploying a Policy to an Identity Vault" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Compare	Compares the Schema Map policy in Designer to the Schema Map policy in the Identity Vault. For more information, see "Using the Compare Feature When Deploying" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Open With > Designer Built-in Editor	Launches the Schema Map editor.
Open With > NetIQ XML Editor	Launches the XML editor. For more information, see "The NetIQ XML Editor" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Open With > Text Editor	Launches the text editor.
Delete	Deletes the selected Schema Map policy.
Properties	Allows you to rename the Schema Map policy.

Policy Flow View Additional Options

There are additional options to manage the Schema Map policy in the Policy Flow view. Right-click the Schema Map policy in the Policy Flow view to see the additional options.

Table 5-3 *Enter Table Title Here*

Option	Description
Add Policy > DirXML Script	Adds a new Schema Map policy by using DirXML Script.
Add Policy > XSLT	Adds a new Schema Map policy by using XSLT.
Add Policy > Schema Map	Adds a new Schema Map policy containing no information.
Add Policy > Link to Existing	Allows you to browse and select an existing Schema Map policy to link to the current Schema Map policy.
Add Policy > Copy Existing	Allows you to browse to and select an existing Schema Map policy to copy to the current Schema Map policy.
Edit Policy > Schema Map	Launches the Schema Map editor. For more information, see "Editing a Schema Map Policy" on page 79.
DirXML Script Tracing	Enables DirXML Script tracing on the Schema Map policy.
Simulate	Tests the Schema Map policy. For more information, see "Testing Schema Map Policies" on page 86.
Live > Import	Imports and existing Schema Map policy from the Identity Vault. For more information, see "Importing Channels, Policies, and Schema Items from the Identity Vault" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Deploy	Deploys the selected Schema Map policy into the Identity Vault. For more information, see "Deploying a Policy to an Identity Vault"*Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Compare	Compares the selected Schema Map policy to a Schema Map policy in the Identity Vault. For more information, see "Using the Compare Feature When Deploying" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Option	Description
Live > Driver Configuration > Import Attribute	Allows you to import attributes from the Identity Vault and compare the attributes from the Identity Vault to what is in Designer. For more information, see "Importing Channels, Policies, and Schema Items from the Identity Vault" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Driver Configuration > Deploy Attributes	Allows you to deploy attributes from Designer into the Identity Vault and compare the attributes from Designer with the attributes in the Identity Vault. For more information, see "Deploying a Policy to an Identity Vault" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Driver Configuration > Compare Attributes	Allows you to compare attributes from the selected Schema Map policy to attributes in the Identity Vault. For more information, see "Using the Compare Feature When Deploying" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Live > Driver Status	Displays the status of the driver.
Live > Start Driver	Starts the driver.
Live > Stop Driver	Stops the driver.
Live > Restart Driver	Restarts the driver.
Delete All Set Policies	Deletes all policies in the selected policy set.
Remove All Set Policies	Removes all policies from the selected policy set, but does not delete the existing policies.

Policy Set View Additional Options

There are additional options to manage the Schema Map policy in the Policy Set view. Right-click the Schema Map policy in the Policy Set view to see the additional options.

Table 5-4 *Policy Set View Options*

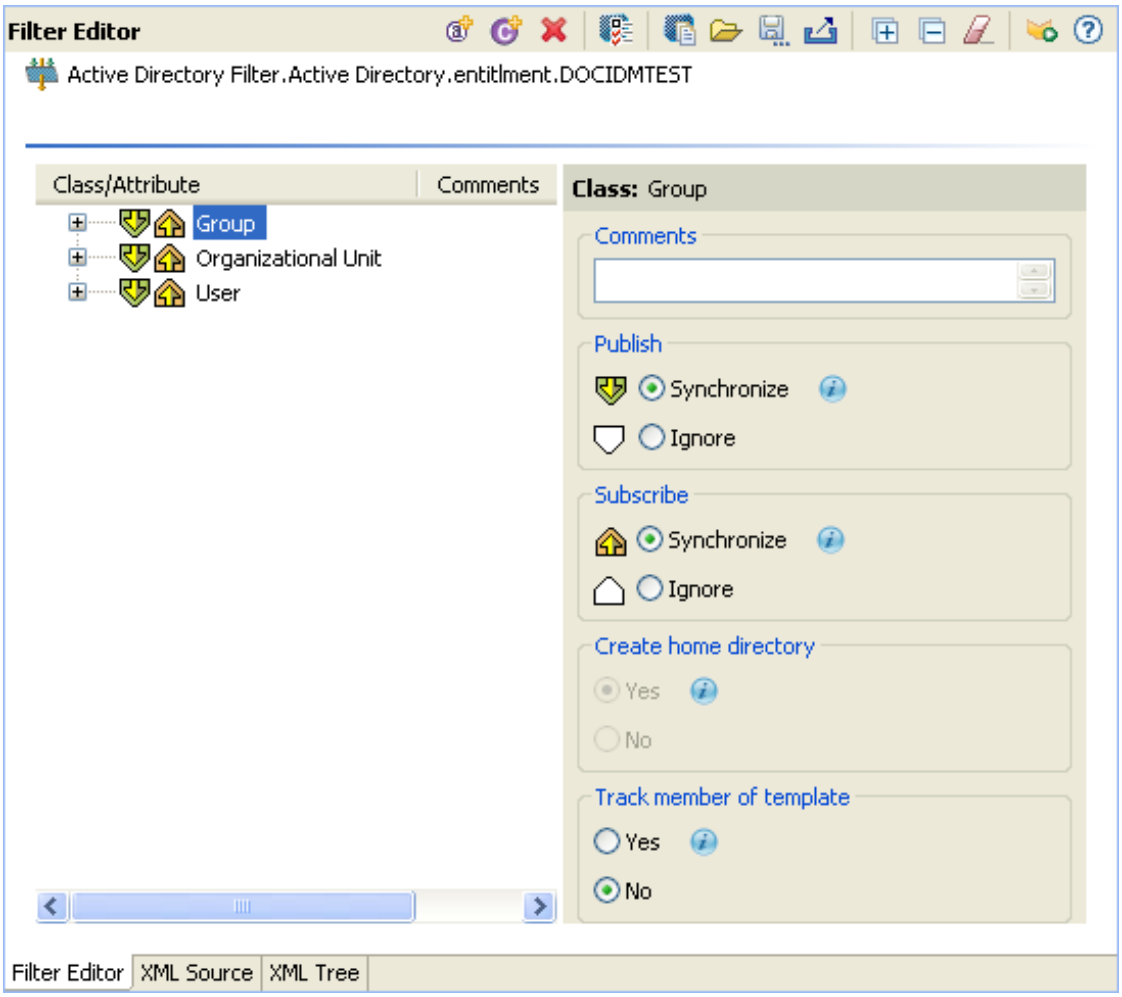
Option	Description
Edit	Launches the Schema Map editor. For more information, see "Editing a Schema Map Policy" on page 79.
Copy	Creates a copy of the Schema Map policy.
Save As	Saves the Schema Map policy as a `.xml` file.
Simulate	Tests the Schema Map policy. For more information, see "Testing Schema Map Policies" on page 86.

Option	Description
Remove	Removes the Schema Map policy from the policy set, but does not delete the Schema Map policy from the Identity Vault.
Link to Existing Policy	Allows you to browse to another Schema Map policy and link it into the existing policy.
Move up	Moves the Schema Map policy up in the execution order of the policy.
Move down	Moves the Schema Map policy down in the execution order of the policy.
Export to Configuration File	Saves the Schema Map policy as a `.xml` file.
Live > Deploy	Deploys the Schema Map policy into the Identity Vault.
Live > Compare	Compares the Schema Map policy in Designer to the Schema Map policy in the Identity Vault.
Delete	Deletes the selected Schema Map policy.
Properties	Allows yo to rename the Schema Map policy.

6 Controlling the Flow of Objects with the Filter

The Filter editor allows you to manage the filter. In the Filter editor, you define how each class and attribute should be handled by the Publisher and Subscriber channels.

Figure 6-1 *The Filter Editor*



When information is synchronized between connected systems, the connected system can receive the changes or just be notified that a change has occurred. Designer displays this information in the Policy Flow view as **Sync** and **Notify** filters.

If a filter is set to Sync, then the objects modifications are automatically synchronized to the connected system. If the filter is set to Notify, then the object modification is reported to the OpenText Identity Manager engine, but the object is not automatically synchronized. For more information, see "Changing the Filter Settings" on page 98.

This section includes the following topics:

- “Using the Filter Editor” on page 92
- “Editing the Filter” on page 96
- “Testing the Filter” on page 102
- “Exporting and Importing Filter Files” on page 102
- “Adding Comments to Classes and Attributes” on page 103
- “Viewing the Filter in XML” on page 103
- “Deploying the Filter” on page 103
- “Additional Filter Options” on page 103

Using the Filter Editor

The Filter editor allows you to edit filter policies. This section includes the following topics:

- “Accessing the Filter Editor” on page 92
- “Navigating the Filter Editor” on page 95
- “Understanding the Filter Editor Toolbar” on page 95

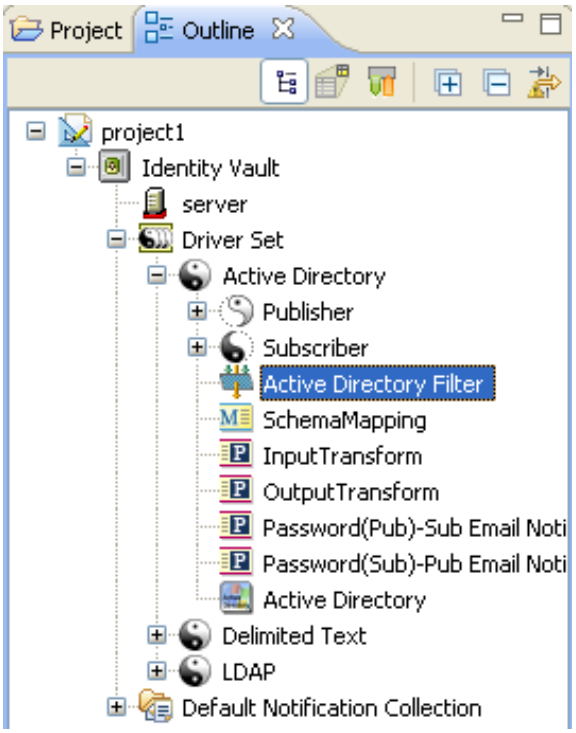
Accessing the Filter Editor

The Filter editor allows you to edit the filter. There are three different ways to access the Filter editor:

- “Model Outline View” on page 92
- “Policy Flow View” on page 93
- “Policy Set View” on page 94

Model Outline View

1. In the Outline view, select the *Show Model Outline* icon .
2. In the Model Outline, open the driver for which you want to manage a filter.

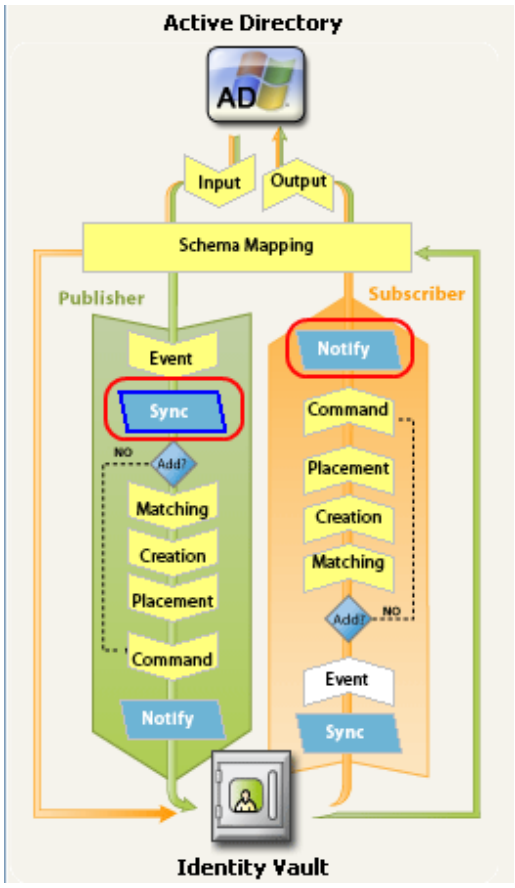


3 Double-click the Filter object (or right-click it and select **Edit**) to launch the Filter editor.

Policy Flow View

1 In the Outline view, select the **Show Policy Flow** icon.

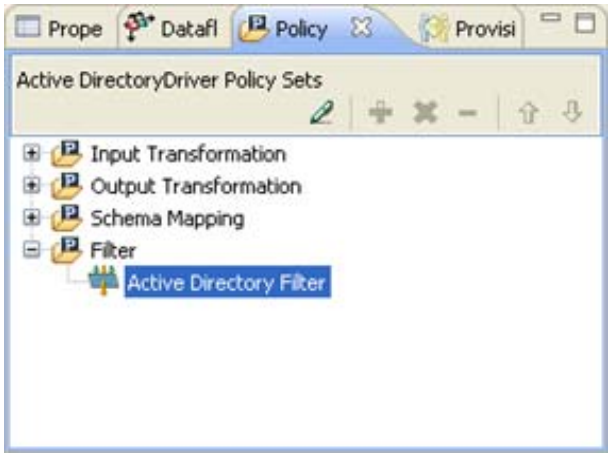
NOTE: We recommend you to import the policy from outline view.



2 In the Policy Flow, double-click the **Sync** icon or the **Notify** objects (or Right-click and select **Edit Policy > Filter**) to launch the Filter editor.

Policy Set View

1 Double-click the filter object in the Policy Set view.



Navigating the Filter Editor

The Filter Editor uses standard point-and-click navigation. However, it also provides keyboard-based navigation options as described in Table 7-1.

NOTE: The Filter Editor lets you order the classes/attributes as needed:

- Click the header bar above the class/attribute list to switch between ascending and descending order. This sorts both the classes and the attributes within the classes.
- Click and drag individual classes or attributes to create a custom order.

Table 6-1 *Filter Editor Keyboard Support*

Keystroke	Description
Up-arrow	Moves the cursor up in the Filter editor.
Down-arrow	Moves the cursor down in the Filter editor.
Left-arrow	Collapses the information displayed.
Right-arrow	Expands the information displayed.
Insert	Adds a class.
Ctrl+Insert	Adds an attribute.
Delete	Deletes the selected items.
Esc	Exits the edit mode.
Ctrl+A	Selects all classes and attributes in the Filter editor.

Understanding the Filter Editor Toolbar

The Filter editor includes a toolbar that provides access to the following features. Each of these features, along with options to Undo and Redo recent actions, is also available from a drop-down menu by right-clicking in the Filter Editor.

Tool	Description
	Add Attributes opens the Schema Browser so you can select attributes from the selected class to add to the filter policy. For more information, see "Adding an Attribute" on page 97.
	Add Classes opens the Schema Browser so you can select classes from the Identity Vault schema to add to the filter policy. For more information, see "Adding a Class" on page 97.
	Delete deletes the selected attributes and classes from the filter policy.
	Default Attribute Settings lets you define default values for all attributes added to the filter policy. For more information, see "Setting Default Values for Attributes" on page 97.
	Copy an Existing Filter lets you copy the filter policy from another Designer object. For more information, see "Copying an Existing Filter" on page 97.

Tool	Description
	Import Filter imports an existing filter policy from a previously saved XML file. For more information, see "Importing a Filter File" on page 102.
	Export Filter saves the current filter policy to an XML file. For more information, see "Exporting a Filter File" on page 102.
	Deploy Filter deploys the filter policy to a live OpenText Identity Manager environment. For more information, see "Deploying the Filter" on page 103.
	Expand All expands all Class/Attribute groups in the filter policy.
	Collapse All collapses all Class/Attribute groups in the filter policy.
	Clear Filter deletes all class and attribute entries from the filter policy.
	Launch Policy Simulator launches the Policy Simulator. For more information, see Chapter 8, "Testing Policies with the Policy Simulator," on page 143.
	Help launches the context-sensitive help for the Filter editor.

Editing the Filter

The Filter editor allows you to create and edit the filter. It provides the following primary tasks:

- "Removing or Adding Classes and Attributes" on page 96
- "Modifying Multiple Attributes" on page 97
- "Copying an Existing Filter" on page 97
- "Setting Default Values for Attributes" on page 97
- "Changing the Filter Settings" on page 98

Removing or Adding Classes and Attributes

By removing or adding classes and attributes, you determine the objects that synchronize between the connected data store and the Identity Vault.

- "Removing a Class or Attribute" on page 96
- "Adding a Class" on page 97
- "Adding an Attribute" on page 97

Removing a Class or Attribute

If you do not want a class or an attribute to synchronize, the best practice is to completely remove the class or the attribute from the filter. To remove attributes and classes from the filter, do one of the following:

- Right-click the class or attribute you want to remove, then select **Delete**.

- Select the class or attribute you want to remove, then click **Delete** .
- Click **Clear Filter** to delete all classes and attributes from the filter.

Adding a Class

1. Click **Add Classes**.

   You can also right-click in the Filter editor, then select **Add Classes**.
2. Browse and select the class you want to add, then click **OK**.
3. Change the options to synchronize the information.
4. To save the changes, click **File > Save**.

Adding an Attribute

1. Click **Add Attributes**.

   You can also right-click in the Filter editor, then select **Add Attribute**.
2. Browse and select the attribute you want to add, then click **OK**.
3. Change the options to synchronize the information.
4. To save the changes, click **File > Save**.

Modifying Multiple Attributes

The Filter editor allows you to modify more than one attribute at a time. Press the Ctrl key and select multiple attributes; when the option changes, it is changed for all of the selected attributes.

Copying an Existing Filter

You can copy an existing filter from another driver and use it in the driver you are currently working with.

1. Click **Copy an Existing Filter**.

   You can also right-click in the Filter editor, then select **Copy an Existing Filter**.
2. Browse to and select the filter object you want to copy, then click **OK**.

   If you have more than one Identity Vault in your project, you can copy filters from the other Identity Vaults. When you are browsing to select the other object, you can browse to the other Identity Vault and use a filter stored there.

Setting Default Values for Attributes

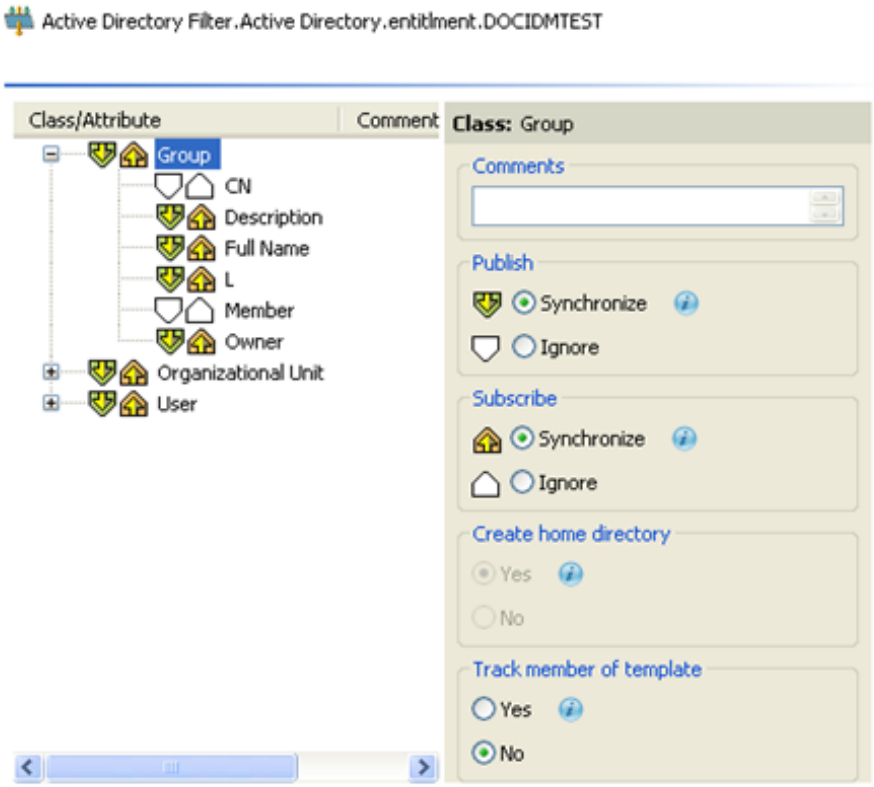
You can define the default values for new attributes when they are added to the filter.

1. Click **Default Attribute Settings**.
2. Select the options you want new attributes to have, then click **OK**.

Changing the Filter Settings

The Filter editor gives you the option of changing how information is synchronized between the Identity Vault and the connected system. The filter has different settings for classes and attributes.

1 In the Filter editor, select a class.



2 Change the filter settings for the selected class.

See Table 6-2 on page 98 for information on each of the class settings available in the Filter Editor.

3 In the Filter Editor, select an attribute.

4 Change the filter settings for the selected attribute, then click **Save** (in the Designer toolbar) to save the changes.

See Table 6-3 on page 99 for information on each of the attribute settings available in the Filter Editor.

Table 6-2 *Filter Editor Class Settings*

Options	Definitions
Publish	◆ **Synchronize:** Allows the class to synchronize from the connected system into the Identity Vault. ◆ **Ignore:** Does not synchronize the class from the connected system into the Identity Vault.

Options	Definitions
Subscribe	◆ **Synchronize:** Allows the class to synchronize from the Identity Vault into the connected system. ◆ **Ignore:** Does not synchronize the class from the Identity Vault into the connected system.
Create Home Directory	Create Home Directory allows you to create a home directory for a User object in OpenText eDirectory. The option only works for OpenText eDirectory. ◆ **Yes:** Automatically creates home directories. ◆ **No:** Does not create home directories.
Track Member of Template	◆ **Yes:** Determines whether or not the Publisher channel maintains the Member of Template attribute when it creates objects from a template. ◆ **No:** Does not track the Member of Template attribute. When a User object is created using an OpenText eDirectory Template object, the OpenText eDirectory driver maintains the Member of Template attribute, if the Track Member of Template option is selected. The option only works for OpenText eDirectory.

Table 6-3 *Filter Editor Attribute Settings*

Options	Definitions
Publish	◆ **Synchronize:** Changes to this object are reported and automatically synchronized. ◆ **Ignore:** Changes to this object are neither reported nor automatically synchronized. ◆ **Notify:** Changes to this object are reported, but not automatically synchronized. ◆ **Reset:** Resets the object value to the value specified by the opposite channel. (You can set this value on either the Publisher channel or Subscriber channel, not both.) The Reset option makes a data store the authoritative source of information. For example, if employee addresses should only be changed in the HR database, then set the Reset option in the filter for this attribute. When an address is changed in the e-mail system and sent to the HR database, the filter sends the information from the HR database back to the e-mail system and the employee's address is not changed.

Options	Definitions
Subscribe	◆ **Synchronize:** Changes to this object are reported and automatically synchronized. ◆ **Ignore:** Changes to this object are neither reported nor automatically synchronized. ◆ **Notify:** Changes to this object are reported, but not automatically synchronized. ◆ **Reset:** Resets the object value to the value specified by the opposite channel. (You can set this value on either the Publisher channel or Subscriber channel, not both.) The Reset option makes a data store the authoritative source of information. For example, if employee addresses should only be changed in HR database, then set the Reset option in the filter for this attribute. When an address is changed in the e-mail system and sent to the HR database, the filter sends the information from the HR database back to the e-mail system and the employee's address is not changed.

Options	Definitions
Merge Authority	◆ **Default:** If an attribute is not being synchronized in either channel, no merging occurs. If an attribute is being synchronized in one channel and not the other, then all existing values on the destination for that channel are removed and replaced with the values from the source for that channel. If the source has multiple values and the destination can only accommodate a single value, then only one of the values is used on the destination side. If an attribute is being synchronized in both channels and both sides can accommodate only a single value, the connected application acquires the Identity Vault values unless there is no value in the Identity Vault. If this is the case, the Identity Vault acquires the values from the connected application (if any). If an attribute is being synchronized in both channels and only one side can accommodate multiple values, the single-valued side's value is added to the multi-valued side if it is not already there. If there is no value on the single side, you can choose the value to add to the single side. This is always valid behavior. ◆ **Identity Vault:** Behaves the same way as the default behavior if the attribute is being synchronized on the Subscriber channel and not on the Publisher channel. This is valid behavior when synchronizing on the Subscriber channel. ◆ **Application:** Behaves the same as the default behavior if the attribute is being synchronized on the Publisher channel and not on the Subscriber channel. This is valid behavior when synchronizing on the Publisher channel. ◆ **None:** No merging occurs regardless of synchronization.
Optimize modifications to the Identity Vault	◆ **Yes:** Changes to this attribute are examined on the Publisher channel to determine the minimal change made in the Identity Vault. ◆ **No:** Changes are not examined. When an operation is a Modify on the Publisher channel, the OpenText Identity Manager engine examines the current state of the object in the Identity Vault and changes the Modify to update only the values that are changing. For example, if an object has attributes of a, b, c, and d and the Publisher channel receives a Modify event to remove all existing values and add a, b, d, and e, the optimize process knows that the minimal change is to remove d and add e. Using this option can take a long time to process events on attributes that have more than 1,000 values.

Options	Definitions
Perform Out of Band Sync (Subscriber)	This option is available only for filter attributes on the subscriber channel and is set to No by default. ◆ **Yes:** Allows you to assign a higher priority to the selected attribute and have it processed before the other attributes that are in the queue. ◆ **No:** Attributes are processed in the order of their occurrence.

Testing the Filter

Designer comes with a tool called the Policy Simulator, which allows you to test policies without implementing them in a production environment. You can launch the Policy Simulator through the Filter editor to test your policy after you have modified it.

1 Click **Launch Policy Simulator** .

2 Select **To Identity Vault** or **From Identity Vault** as the simulation point of the filter.

For more information on the Policy Simulator, see Chapter 8, "Testing Policies with the Policy Simulator," on page 143.

Exporting and Importing Filter Files

Designer allows you to Import an XML filter file from a particular point on the file system to the filter editor. It also allows you to Export an XML filter file to a particular location on the file system.

- "Exporting a Filter File" on page 102
- "Importing a Filter File" on page 102

Exporting a Filter File

1 Select **Export Filter** .

2 In the Export Filter dialog box, specify a file name and location for the XML filter file, then click *Save*.

Importing a Filter File

1 Select **Import Filter** .

2 In the Import Filter File dialog box, browse to the filter file you want to import, then click *Open*.

Specify whether you want to append the imported filter rules to the existing filter rules, or replace the existing filter rules with the imported filter rules.

NOTE: Both the Import and Export features enable the user to export filter editor documents and re-import them if required, thereby avoiding the need to redo the entire task of adding classes and attributes and assigning their properties.

Adding Comments to Classes and Attributes

Filter Editor lets you add additional comments to the classes and attributes in the filter. These comments are visible in the Filter Editor, and in Designer's generated documentation for the project.

1. In the Filter Editor, select the class or attribute to which you want to add a comment, then type the desired comment in the **Comments** field.

Once entered, the comment is visible in the Comments column next to its associated class or attribute.

Viewing the Filter in XML

Designer enables you to view, edit, and validate the XML by using an XML editor. Click the **XML Source** tab or the **XML Tree** tab to access the XML editor. For more information about the XML editor, see "The NetIQ XML Editor" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Deploying the Filter

To deploy the filter to the live Identity Vault:

1. Click **Deploy Filter** .
2. In the Deployment Summary page, click **Deploy**.

   The Deployment Summary displays Designer's Compare feature so you know what the differences are between Designer's filter and the currently deployed filter, if any. For more information about the Compare feature, see "Using the Compare Feature When Deploying" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
3. In the Deployment Results page, click **OK**.

   The Deployments Results page notes any errors or warnings that occurred during the deployment process.

Additional Filter Options

When you right-click a filter object, there are multiple options presented in the Outline view, the Policy Flow view, and the Policy Set view.

- "Outline View Additional Options" on page 103
- "Policy Flow View Additional Options" on page 104
- "Policy Set View Additional Options" on page 105

Outline View Additional Options

The Outline view offers the following filter-related options. To access them, right-click the filter object in the Outline view.

Table 6-4 Filter Outline View Additional Options

Option	Description
Edit	Launches the Filter editor. For more information, see "Editing the Filter" on page 96.
Save As	Saves the filter as a `.xml` file.
Simulate	Launches the Policy Simulator. For more information, see "Testing the Filter" on page 102.
Export to Configuration File	Saves the filter as a `.xml` file.
Live > Deploy	Deploys the filter into the Identity Vault.
Live > Compare	Compares the filter with and existing filter object in the Identity Vault.
Open With > Designer Built-in Editor	Launches the Filter editor. For more information, see "Editing the Filter" on page 96.
Open With > NetIQ XML Editor	Launches the XML editor. For more information, see "The NetIQ XML Editor" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .
Open With > Text Editor	Launches the built-in text editor.

Policy Flow View Additional Options

The Policy Flow view offers the following filter-related options. To access them, right-click the filter object in the Policy Flow view.

Table 6-5 Filter Policy Flow View Additional Options

Option	Description
Edit	Launches the Filter edit. For more information, see "Editing the Filter" on page 96.
Save As	Saves the selected Policy Set as a `.xml` file.
Simulate	Launches the Policy Simulator. For more information, see "Testing the Filter" on page 102.
Live > Import	Allows you to import filter details from the Identity Vault.
Live > Deploy	Allows you to deploy the filter into the Identity Vault.
Live > Compare	Compares the filter to an existing filter in the Identity Vault.
Live > Driver Configuration > Import Attributes	Allows you to import attributes from the Identity Vault and compare the attributes from the Identity Vault to what is in Designer.

Option	Description
Live > Driver Configuration > Deploy Attributes	Allows you to deploy attributes from Designer into the Identity Vault and compare the attributes from Designer with the attributes in the Identity Vault.
Live > Driver Configuration > Compare Attributes	Allows you to compare attributes from the selected Schema Map policy to attributes in the Identity Vault.
Live > Driver Status	Displays the status of the driver.
Live > Start Driver	Starts the driver.
Live > Stop Driver	Stops the driver.
Live > Restart Driver	Restarts the driver.

Policy Set View Additional Options

The Policy Set view offers the following filter-related options. To access them, right-click the filter object in the Policy Set view.

Table 6-6 *Filter Policy Set View Additional Options*

Option	Description
Edit	Launches the Filter editor. For more information, see "Editing the Filter" on page 96.
Save As	Saves the filter as a `.xml` file.
Simulate	Launches the Policy Simulator. For more information, see "Testing the Filter" on page 102.
Export to a Configuration File	Save the filter as a `.xml` file.
Live > Deploy	Allows you to deploy the filter into the Identity Vault.
Live > Compare	Compares the filter to an existing filter in the Identity Vault.

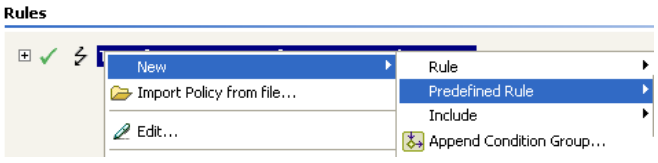
7 Using Predefined Rules

Designer includes 19 predefined rules for common administrative tasks. You can import and use these rules as well as create your own rules. To customize the rules, you must provide information specific to your environment.

- “Accessing the Predefined Rules” on page 107
- “Command Transformation - Create Departmental Container - Part 1 and Part 2” on page 108
- “Command Transformation - Publisher Delete to Disable” on page 110
- “Creation - Require Attributes” on page 112
- “Creation - Publisher - Use Template” on page 114
- “Creation - Set Default Attribute Value” on page 115
- “Creation - Set Default Password” on page 117
- “Event Transformation - Scope Filtering - Include Subtrees” on page 119
- “Event Transformation - Scope Filtering - Exclude Subtrees” on page 120
- “Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn” on page 122
- “Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn” on page 124
- “Matching - Publisher Mirrored” on page 125
- “Matching - Subscriber Mirrored - LDAP Format” on page 127
- “Matching - By Attribute Value” on page 129
- “Placement - Publisher Mirrored” on page 131
- “Placement - Subscriber Mirrored - LDAP Format” on page 133
- “Placement - Publisher Flat” on page 134
- “Placement - Subscriber Flat - LDAP Format” on page 136
- “Placement - Publisher By Dept” on page 138
- “Placement - Subscriber By Dept - LDAP Format” on page 141

Accessing the Predefined Rules

In the Policy Builder, right-click and select **New** > **Predefined Rules** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

1 

Command Transformation - Create Departmental Container - Part 1 and Part 2

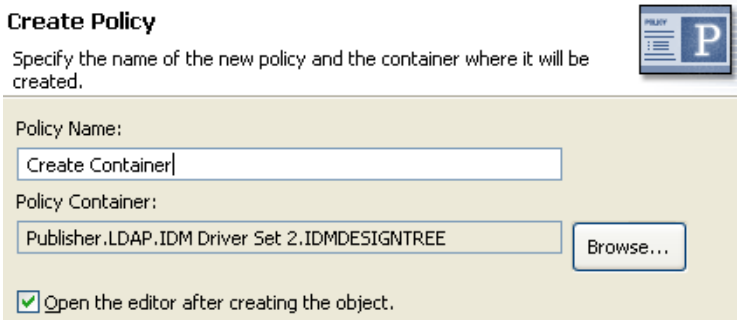
This rule creates a department container in the destination data store, if one does not exist. Implement the rule on the Command Transformation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Command Transformation policy set and importing the predefined rule. If you already have a Command Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 108
- "Importing the Predefined Rule" on page 108
- "How the Rule Works" on page 109

Creating a Policy

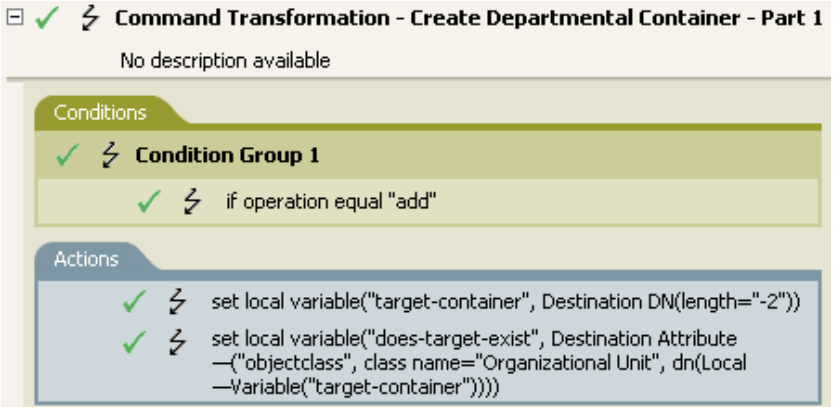
1. From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.
2. Select the Command Transformation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.



6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Command Transformation policy is saved.
9. Continue with Importing the Predefined Rule.

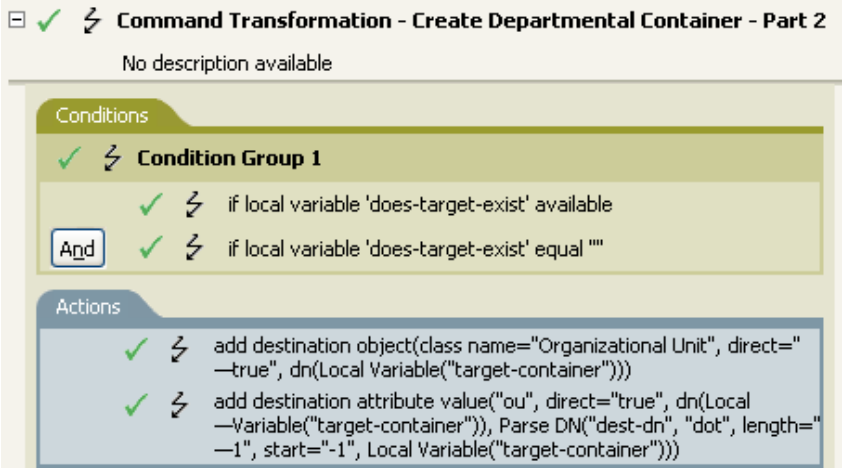
Importing the Predefined Rule

1. Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.
2. Select **Command Transformation - Create Department Container - Part 1**, then click **OK**.



3 Right-click in the Policy Builder and click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

4 Select **Command Transformation - Create Department Container - Part 2**, then click **OK**.



5 Save the rule by clicking **File** > **Save**.

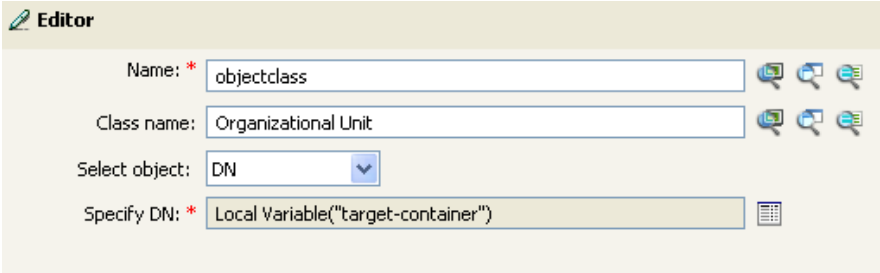
There is no information to change that is specific to your environment.

IMPORTANT: Make sure that the rules are listed in order. Part 1 must be executed before Part 2.

How the Rule Works

This rule is used when the destination location for an object does not exist. Instead of getting a veto because the object cannot be placed, this rule creates the container and places the object in the container.

Part 1 looks for any Add event. When the Add event occurs, two local variables are set. The first local variable is named target-container. The value of target-container is set to the destination DN. The second local variable is named does-target-exist. The value of does-target-exist is set to the destination attribute value of objectclass. The class is set to OrganizationalUnit. The DN of the Organizational Unit is set to the local variable of target-container.



Part 2 checks to see if the local variable does-target-exist is available. It also checks to see if the value of the local variable does-target-exist is set to a blank value. If the value is blank, then an Organizational Unit object is created. The DN of the organizational unit is set to the value of the local variable target-container. It also adds the value for the OU attribute. The value of the OU attribute is set to the local variable of target-container. It uses the source format as the destination DN and the destination format is dot format.

Command Transformation - Publisher Delete to Disable

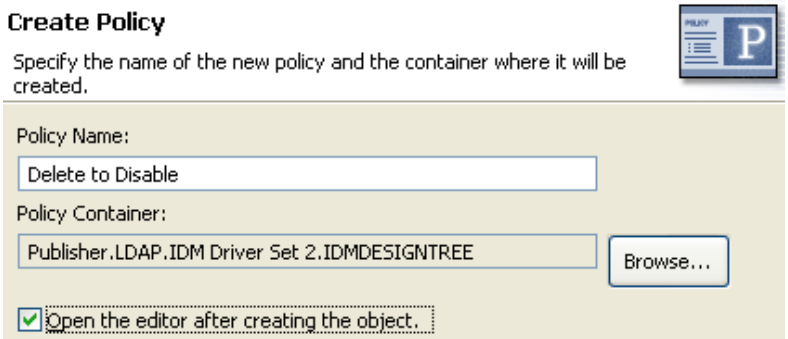
This rule transforms the Delete event for a user object into disabling the user object. Implement the rule on the Command Transformation policy in the driver. The rule needs to be implemented on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Command Transformation policy set and importing the predefined rule. If you already have a Command Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 110
- "Importing the Predefined Rule" on page 111
- "How the Rule Works" on page 112

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Publisher channel.
2. Select the Command Transformation policy set in the Policy Set view, then click **Create or add a new policy to the policy set** icon to create a new policy.
3. Select **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Command Transformation policy is saved.

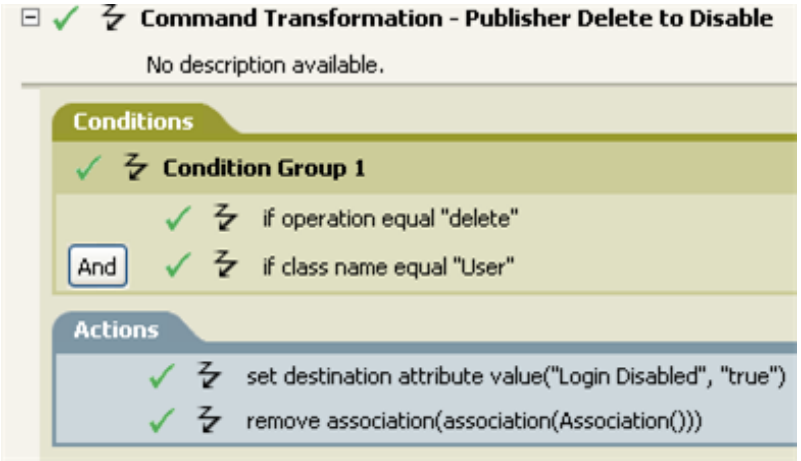
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Command Transformation - Publisher Delete to Disable**, then click **OK**.

3 Expand the predefined rule.



4 Save the rule by clicking **File** > **Save**.

There is no information to change in the rule that is specific to your environment.

How the Rule Works

This rule is used when a Delete event occurs in the connected data store. Instead of the user object being deleted in the Identity Vault, the User object is disabled. Anytime a Delete event occurs for a User object, the destination attribute value of Login Disabled is set to True and the association is removed from the User object. The User object can no longer log in to the OpenText eDirectory tree, but the User object was not deleted.

Creation - Require Attributes

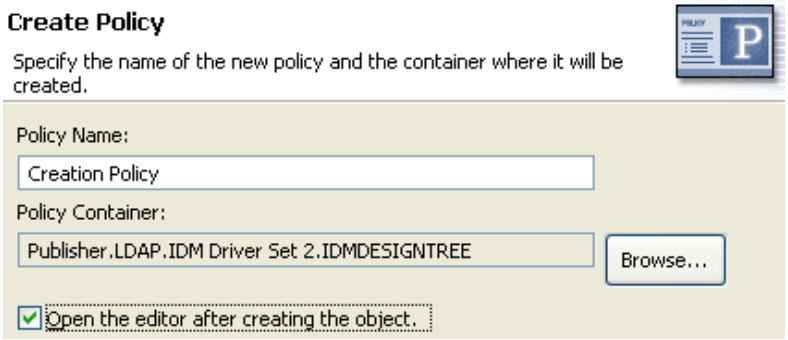
This rule does not allow user objects to be created unless the required attributes are populated. Implement the rule on the Creation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Creation policy set and importing the predefined rule. If you already have a Creation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 112
- "Importing the Predefined Rule" on page 113
- "How the Rule Works" on page 113

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.
2. Select the Creation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.



6. Select **Open Editor** after creating policy, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`” Click **Yes**. The Policy Builder is launched and the new Creation policy is saved.

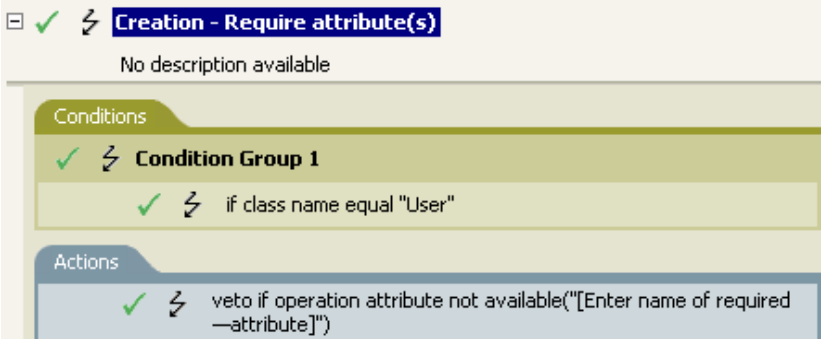
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder and click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Creation - Require attributes**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 In the **Specify Name** field, browse to and select the attributes you require for a User object to be created, then click **OK**.

6 Click **OK**.

7 Save the rule by selecting **File** > **Save**.

How the Rule Works

This rule is used when your business processes require a user to have specific attributes populated when the user object is created. When a user object is created, the rule vetoes the creation of the object unless the required attributes are provided. You can have one or more required attributes.

If you want more than one required attribute, right-click the **Actions** tab and select **Append Action**. Select **veto if operation attribute not available**, then browse to the attribute you want to require.

Creation - Publisher - Use Template

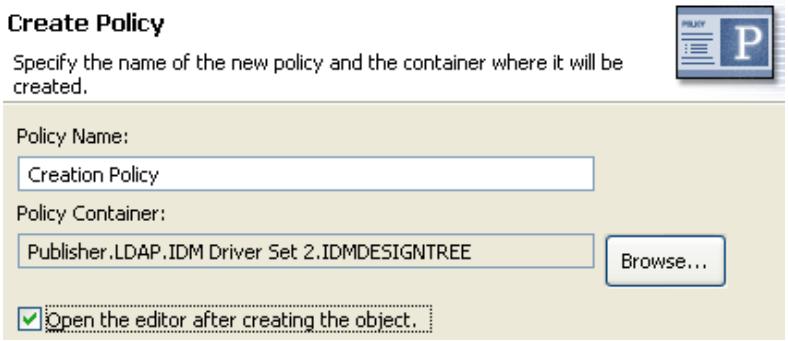
This rule allows the use of a OpenText eDirectory template object during the creation of a User object. Implement the rule on the Publisher Creation policy in the driver. You can implement the rule only on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Creation policy set and importing the predefined rule. If you already have a Creation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 114
- "Importing the Predefined Rule" on page 114
- "How the Rule Works" on page 115

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Publisher channel.
2. Select the Creation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set icon** to create a new policy.
3. Select **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.

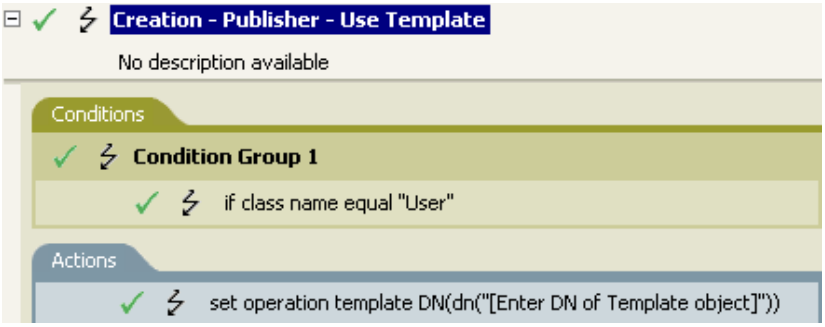


6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Creation policy is saved.
9. Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1. Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.
2. Select **Creation - Publisher - Use Template**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter DN of Template object]** from the **Enter DN** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, click the browse icon, browse to and select the template object, then click **OK**.

10 Click **Finish**.

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to use a template object to create a user in the Identity Vault. If you have attributes that are the same for different users, using the template saves time. You fill in the information in the template object, and when the User object is created, OpenText Identity Manager calls the template and uses that to create the User object.

During the creation of User objects, the rule performs the action of the set operation template DN. The action calls the template object and creates the User object with the information in the template.

Creation - Set Default Attribute Value

This rule allows you to set default values for attributes that are assigned during the creation of User objects. Implement the rule on the Subscriber Creation policy or Publisher Creation policy in the driver.

There are two steps involved in using the predefined rules: creating a policy in the Creation policy set and importing the predefined rule. If you already have a Creation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 116
- "Importing the Predefined Rule" on page 116
- "How the Rule Works" on page 117

Creating a Policy

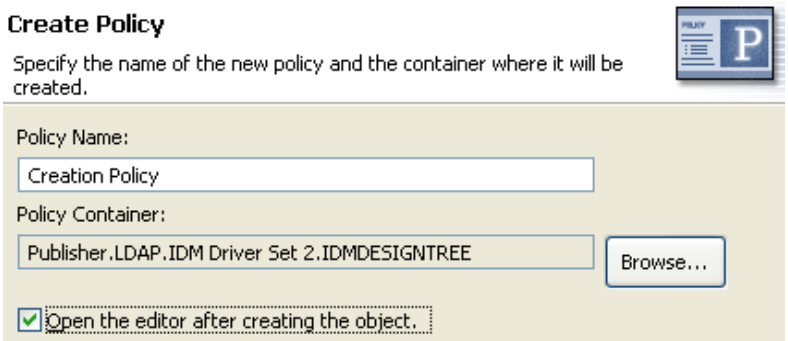
1 From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.

2 Select the Creation policy set in the Policy Set view, then click the **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Creation policy is saved.

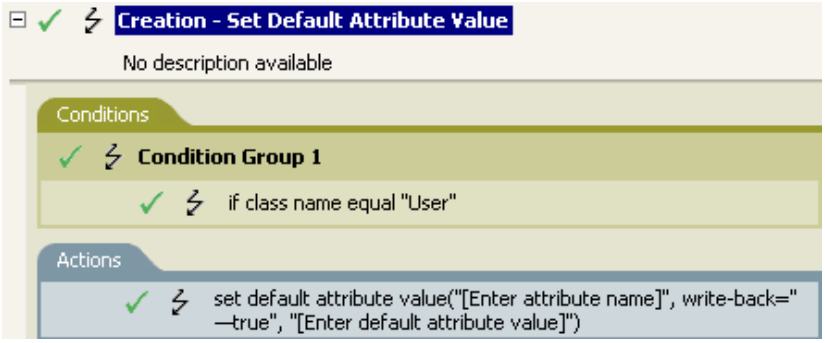
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Creation - Set Default Attribute Value**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 In the **Specify attribute name** field, click the browse icon, then browse to and select the attribute you want to create.

6 Click the **Edit the value list** icon to launch the Argument Value List Builder.

7 Select the type of data you want the value to be.

8 Delete **[Enter default attribute value]**, then click the **Edit the arguments** icon to launch the Argument Builder.

9 Create the value for the attribute in the Argument Builder, then click **OK**.

10 Click **Finish**.

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to create a User object with default attributes and values. When a User object is created, the rule sets the attribute and the value for that attribute.

If you want more than one attribute value defined, right-click the **Actions** tab and click **Append Action**. Select the action, set the default attribute value, and follow Step 1 on page 116 through Step 11 on page 117 to assign the value to the attribute.

Creation - Set Default Password

During the creation of user objects, this rule sets a default password for user objects. Implement the rule on the Creation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Creation policy set and importing the predefined rule. If you already have a Creation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 117
- "Importing the Predefined Rule" on page 118
- "How the Rule Works" on page 118

Creating a Policy

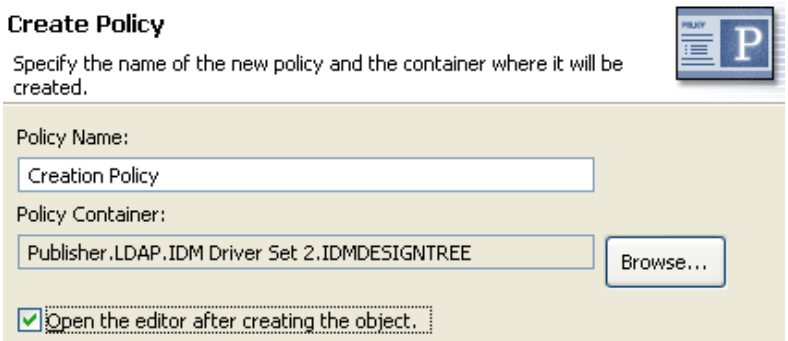
1 From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.

2 Select the Creation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Creation policy is saved.

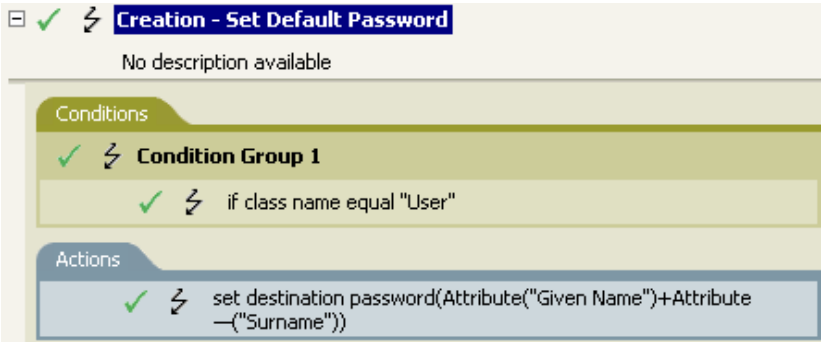
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Creation - Set Default Password**, then click **OK**.

3 Expand the predefined rule.



4 Save the rule by clicking **File** > **Save**.

There is no information to change in the rule that is specific to your environment.

How the Rule Works

This rule is used when you want User objects to be created with a default password. During the creation of a User object, the password that is set for the User object is the Given Name attribute plus the Surname attribute of the User object.

You can change the value of the default password by editing the argument. You can use the Argument Builder to set the password to any other value you want.

Event Transformation - Scope Filtering - Include Subtrees

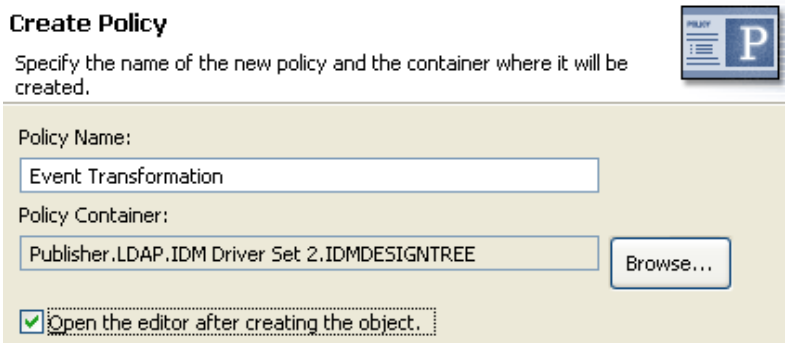
This rule excludes all events that occur except for the specific subtree. Implement the rule on the Event Transformation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Event Transformation policy set and importing the predefined rule. If you already have an Event Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 119
- "Importing the Predefined Rule" on page 119
- "How the Rule Works" on page 120

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.
2. Select the Event Transformation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.

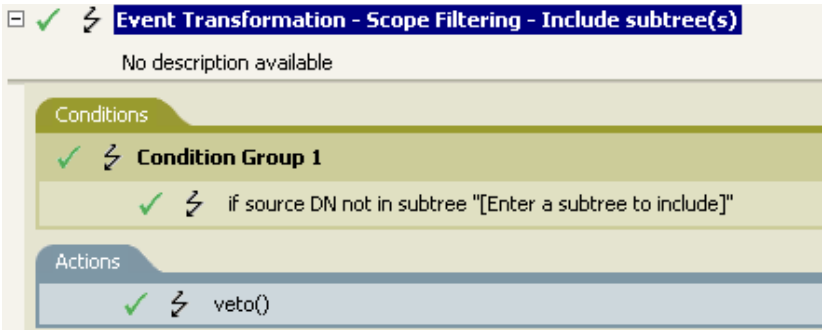


6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Event Transformation policy is saved.
9. Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1. Right-click in the Policy Builder, then select **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.
2. Select **Event Transformation - Scope Filtering - Include subtrees**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 Delete **[Enter a subtree to include]** in the **Value** field.

6 Click the browse button to browse the Identity Vault for the part of the tree you were you want events to synchronize, then click **OK**.

7 Click **OK**.

8 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to exclude part of the Identity Vault from synchronizing. It allows you to synchronize some objects and not other objects, without using the Filter. When an event occurs anywhere but in that specific part of the Identity Vault, it is vetoed.

Event Transformation - Scope Filtering - Exclude Subtrees

This rule excludes all events that occur in a specific subtree. Implement the rule on the Event Transformation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Event Transformation policy set and importing the predefined rule. If you already have an Event Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 120
- "Importing the Predefined Rule" on page 121
- "How the Rule Works" on page 122

Creating a Policy

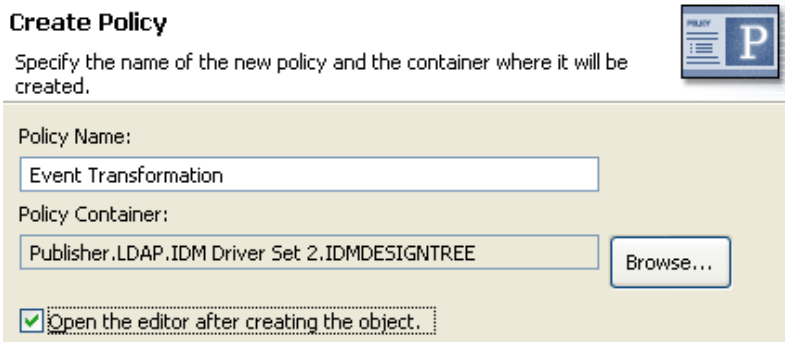
1 From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.

2 Select the Event Transformation policy set in Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Event Transformation policy is saved.

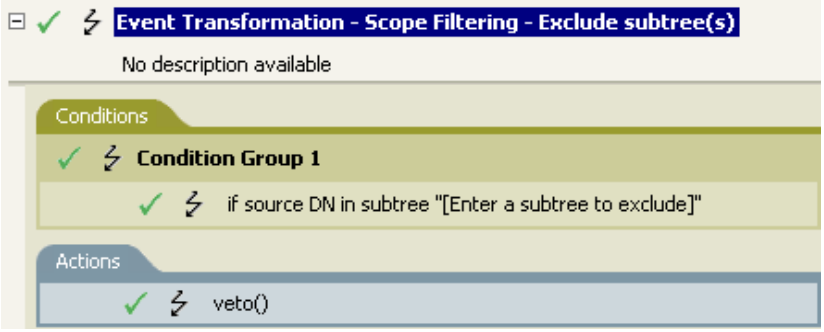
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule**.

2 Select **Event Transformation - Scope Filtering - Exclude subtrees**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 Delete **[Enter a subtree to exclude]** in the **Value** field.

6 Click the browse icon to browse the Identity Vault for the part of the tree where you want to exclude events from synchronizing, then click **OK**.

7 Click **OK**.

8 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to exclude part of the Identity Vault from synchronizing. It allows you to synchronize some objects and not other objects, without using the Filter. When an event occurs in that specific part of the Identity Vault, it is vetoed.

Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn

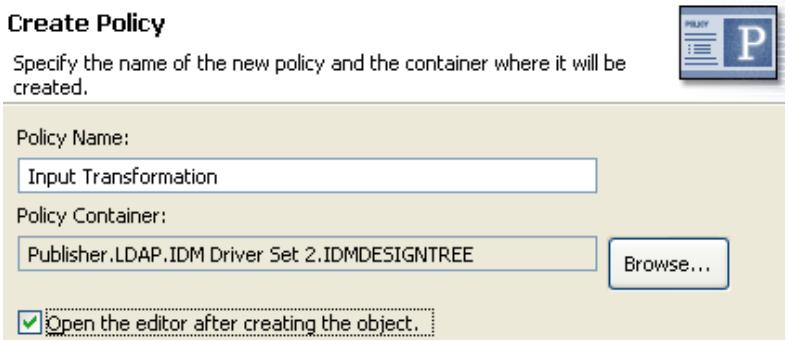
This rule transforms the format of the telephone number when a desired condition is met. Implement the rule on the Input or Output Transformation policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules: creating a policy in the Input or Output Transformation policy set and importing the predefined rule. If you already have an Input or Output Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 122
- "Importing the Predefined Rule" on page 123
- "How the Rule Works" on page 123

Creating a Policy

1. From the Outline view or the Policy Flow view, select a driver.
2. Select the Input or Output Transformation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.



6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`” Click **Yes**. The Policy Builder is launched and the new Input or Output Transformation policy is saved.

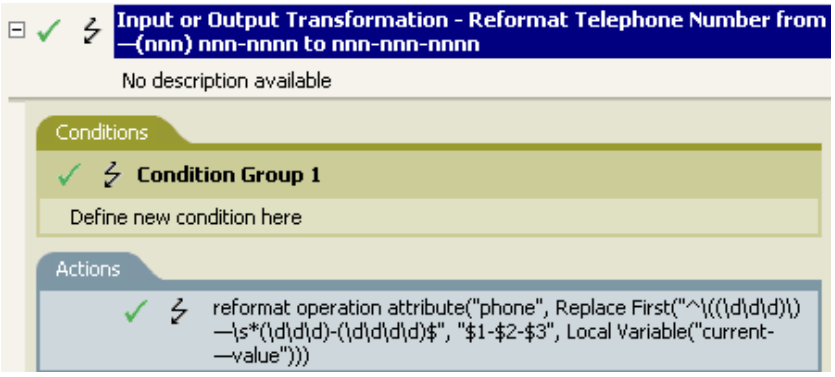
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 Define the condition you want to have occur when the telephone number is reformatted.

6 Click **OK**.

7 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to reformat the telephone number. You define the condition that is to be met when the telephone number is reformatted.

Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn

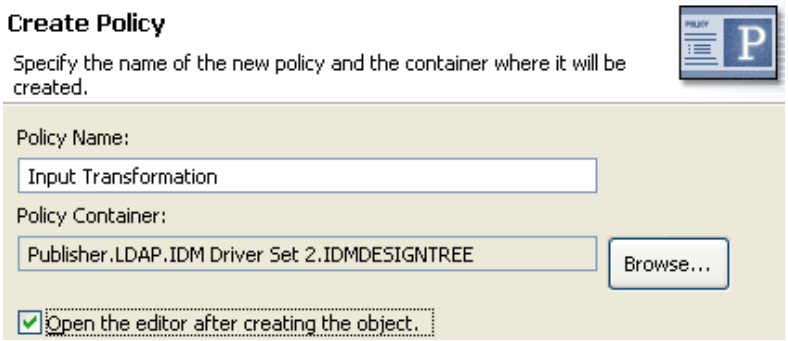
This rule transforms the format of the telephone number when a desired condition is met. Implement the rule on the Input or Output Transformation policy. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules; creating a policy in the Input or Output Transformation policy set and importing the predefined rule. If you already have an Input or Output Transformation policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 124
- "Importing the Predefined Rule" on page 125
- "How the Rule Works" on page 125

Creating a Policy

1. From the Outline view or the Policy Flow view, select a driver.
2. Select the Input or Output Transformation policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.



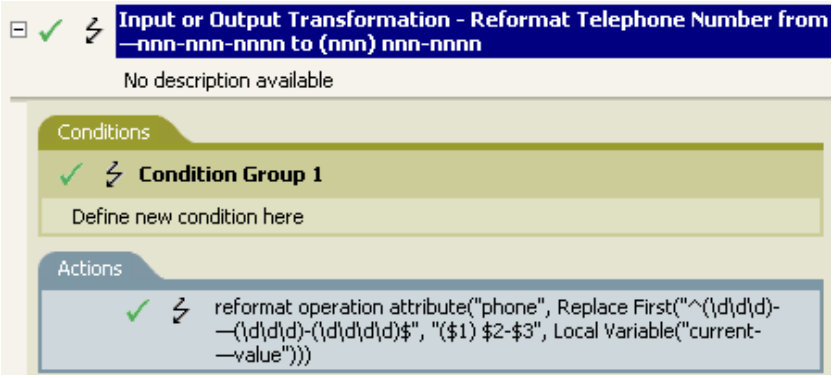
6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Input or Output Transformation policy is saved.
9. Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder and click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Click **Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 Define the condition you want to have occur when the telephone number is reformatted.

6 Click **OK**.

7 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule is used when you want to reformat the telephone number. You define the condition that is to be met when the telephone number is reformatted.

Matching - Publisher Mirrored

This rule matches for objects in the Identity Vault by using the mirrored structure in the data store from a specified point. Implement the rule on the Matching policy in the driver. You can implement the rule only on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Matching policy set and importing the predefined rule. If you already have a Matching policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 126
- "Importing the Predefined Rule" on page 126
- "How the Rule Works" on page 127

Creating a Policy

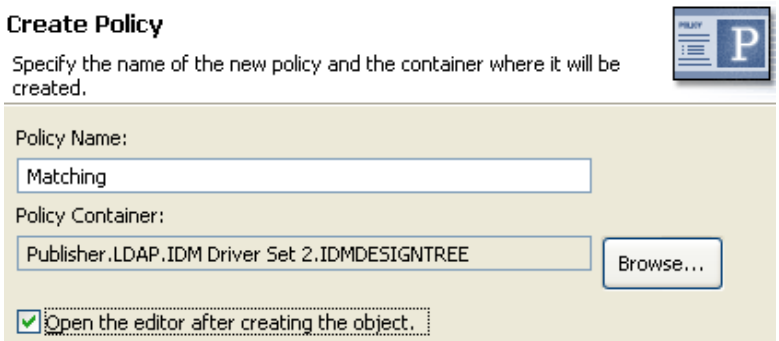
1 From the Outline view or the Policy Flow view, select the Publisher channel.

2 Select the Matching policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Matching policy is saved.

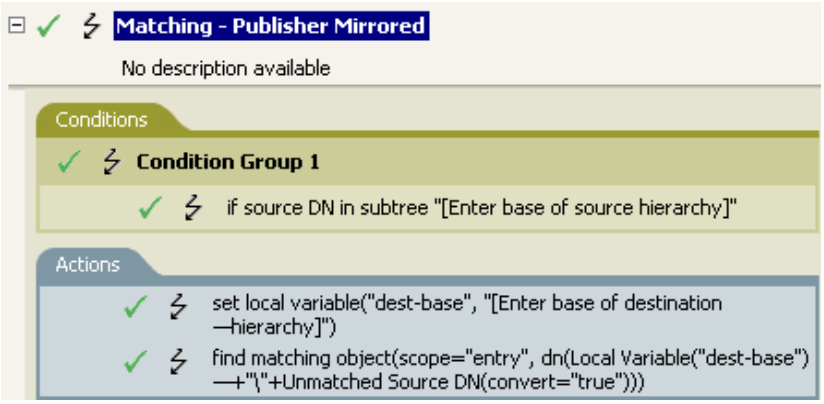
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Matching - Publisher Mirrored**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 In the **Value** field, browse to and select the container in the source hierarchy where you want the matching to start, then click **OK**.

6 Click **OK**.

7 Edit the action by double-clicking the **Actions** tab.

8 Delete **[Enter base of destination hierarchy]** from the **Specify string** field.

9 Click the **Edit the arguments** icon to launch the Argument Builder.

10 Select **Text** in the noun list.

11 Double-click **Text** to add it to the argument.

12 In the editor, click the browse button, browse to the container in the destination hierarchy where you want the source structure to be matched, then click **OK**.

13 Click **Finish**.

14 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule matches for objects in the Identity Vault by using the mirrored structure in the data store from a specified point. When an Add event occurs and the driver checks to see if the object exists, it starts checking at the specific DN in the data store. The driver then sets a local variable of dest-base to be the starting point in the Identity Vault that the structure is mirrored to in the data store. The driver then creates the context it is searching by adding the local variable of dest-base plus a \ and the source DN of the object. It creates the path it is looking for in the slash format.

Matching - Subscriber Mirrored - LDAP Format

This rule matches for objects in the data store by using the mirrored structure in the Identity Vault from a specified point. Implement the rule on the Matching policy in the driver. You can implement the rule only on the Subscriber channel.

There are two steps involved in using the predefined rules: creating a policy in the Matching policy set and importing the predefined rule. If you already have a Matching policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 127
- "Importing the Predefined Rule" on page 128
- "How the Rule Works" on page 129

Creating a Policy

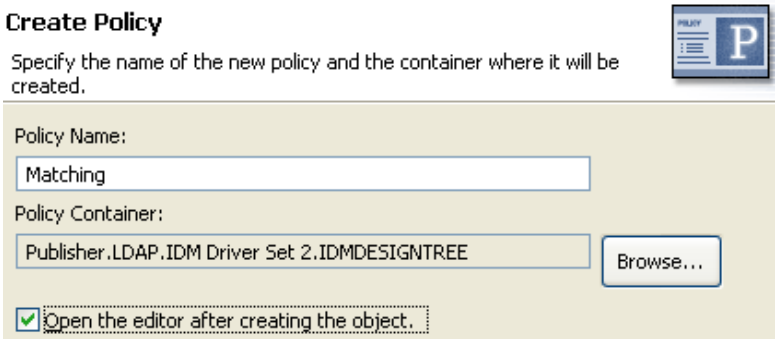
1 From the Outline view or the Policy Flow view, select the Subscriber channel.

2 Select the Matching policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Matching policy is saved.

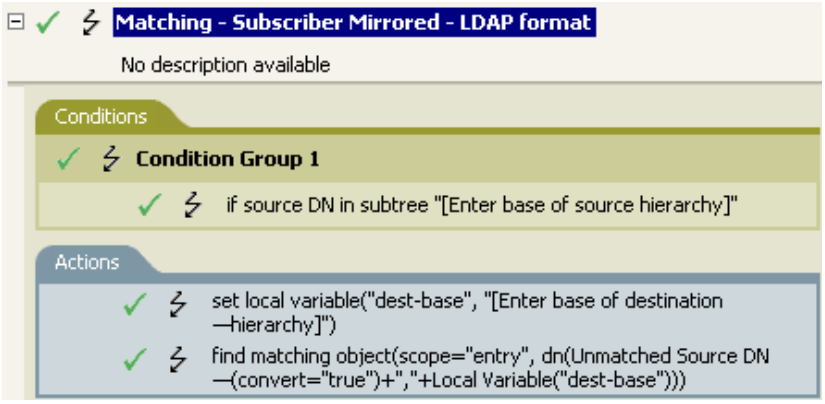
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Matching - Subscriber Mirrored - LDAP format**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 In the **Value** field, browse to and select the container in the source hierarchy where you want the matching to start, then click **OK**.

6 Click **OK**.

7 Edit the action by double-clicking the **Actions** tab.

8 Delete **[Enter base of destination hierarchy]** from the **Specify String** field.

9 Click the **Edit the arguments** icon to launch the Argument Builder.

10 Select **Text** in the noun list.

11 Double-click **Text** to add it to the argument.

12 In the editor, click the browse icon, browse to and select the container in the destination hierarchy where you want the source structure to be matched, then click **OK**.

13 Click **Finish**.

14 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule matches for objects in the data store by using the mirrored structure in the Identity Vault from a specified point. When an Add event occurs and the driver checks to see if the object exists, it starts checking at the specific DN in the Identity Vault. The driver then sets a local variable of dest-base to be the starting point in the data store that the structure is mirrored to in the Identity Vault. The driver then creates the context it is searching by adding the source DN of the object and a local variable of dest-base. It creates the path it is looking for in LDAP format.

Matching - By Attribute Value

This rule matches for objects by specific attribute values. Implement the rule on the Matching policy in the driver. You can implement the rule on either the Subscriber or the Publisher channel or on both channels.

There are two steps involved in using the predefined rules; creating a policy in the Matching policy set and importing the predefined rule. If you already have a Matching policy that you would like to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 129
- "Importing the Predefined Rule" on page 130
- "How the Rule Works" on page 131

Creating a Policy

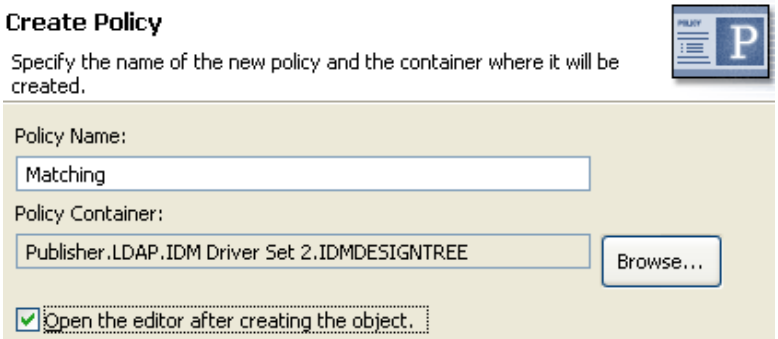
1 From the Outline view or the Policy Flow view, select the Publisher or Subscriber channel.

2 Select the Matching policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Matching policy is saved.

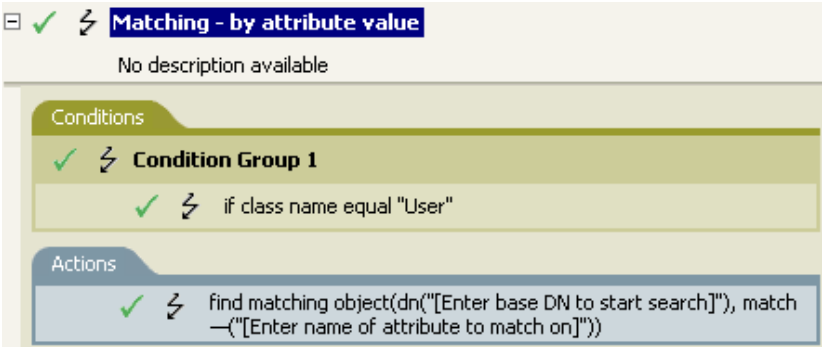
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Matching - by attribute value**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter base DN to start search]** from the **Specify DN** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, click the browse button, browse to and select the container where you want the search to start, then click **OK**.

10 Click **Finish**.

11 In the **Specify Match Attributes** field, click the **Edit the match attributes** icon to launch the Match Attribute Builder.

12 Click the browse button and select the attributes you want to match. You can select one or more attributes to match against, then click **OK**.

13 Click **Finish**.

14 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule matches for User objects by attributes. When a User object is synchronized, the driver uses the rule to check and see if the specified attributes exist. If the attributes do not exist, a new User object is created.

Placement - Publisher Mirrored

This rule places objects in the Identity Vault by using the mirrored structure in the data store from a specified point. Implement the rule on the Placement policy in the driver. You can implement the rule only on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 131
- "Importing the Predefined Rule" on page 132
- "How the Rule Works" on page 132

Creating a Policy

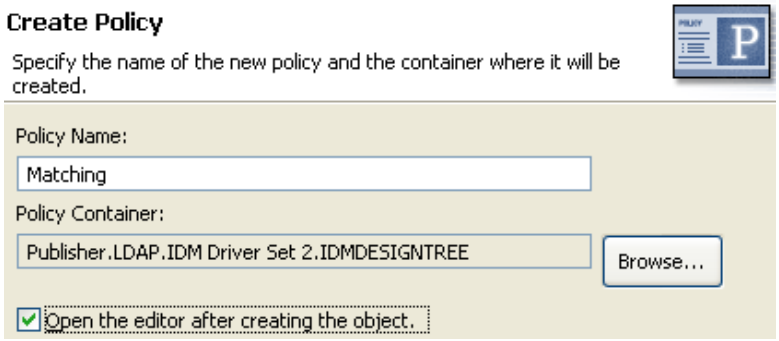
1 From the Outline view or the Policy Flow view, select the Publisher channel.

2 Select the Placement policy set in the policy set, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.

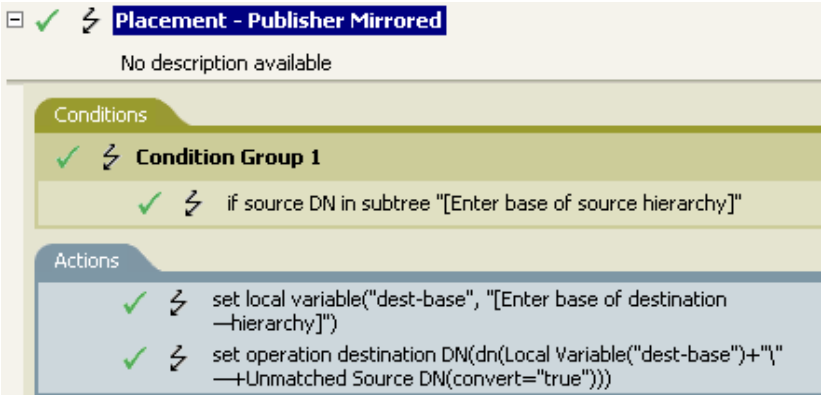
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Placement - Publisher Mirrored**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 In the **Value** field, browse to and select the container in the source hierarchy where you want the object to be acted upon, then click **OK**.

6 Edit the action by double-clicking the **Actions** tab.

7 Delete **[Enter base of destination hierarchy]** from the **Specify String** field.

8 Click the **Edit the arguments** icon to launch the Argument Builder.

9 Select **Text** in the noun list.

10 Double-click **Text** to add it to the argument.

11 In the editor, click the browse button, browse to and select the container in the destination hierarchy where you want the object to be placed, then click **OK**.

12 Click **Finish**.

13 Save the rule by clicking **File** > **Save**.

How the Rule Works

If the User object resides in the source hierarchy, the object is placed in the mirrored structure from the data store. The placement starts at the point that the local variable dest-base is defined. It places the User object in the location of dest-base\unmatched source DN. The rule uses the slash format.

Placement - Subscriber Mirrored - LDAP Format

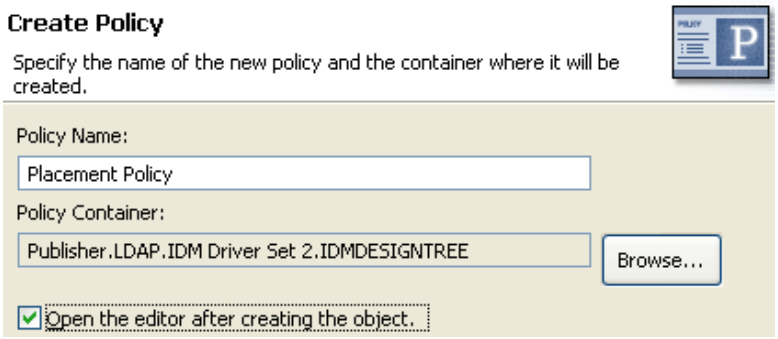
This rule places objects in the data store by using the mirrored structure in the Identity Vault from a specified point. Implement the rule on the Placement policy in the driver. You can implement the rule only on the Subscriber channel.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 133
- "Importing the Predefined Rule" on page 133
- "How the Rule Works" on page 134

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Subscriber channel.
2. Select the Placement policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.

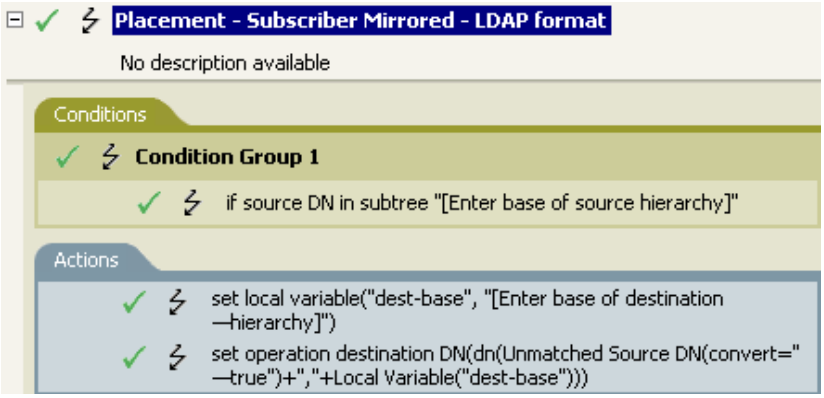


6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.
9. Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1. Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.
2. Select **Placement - Subscriber Mirrored - LDAP format**, then click **OK**.

3 Expand the predefined rule.



4 Edit the condition by double-clicking the **Conditions** tab.

5 In the **Value** field, browse to the container in the source hierarchy where you want the object to be acted upon, then click **OK**.

6 Edit the action by double-clicking the **Actions** tab.

7 Delete **[Enter base of destination hierarchy]** from the **Specify String** field.

8 Click the **Edit the arguments** icon to launch the Argument Builder.

9 Select **Text** in the noun list.

10 Double-click **Text** to add it to the argument.

11 In the editor, click the browse button, browse to the container in the destination hierarchy where you want the object to be placed, then click **OK**.

12 Click **Finish**.

13 Save the rule by clicking **File** > **Save**.

How the Rule Works

If the User object resides in the source hierarchy, then the object is placed in the mirrored structure from the Identity Vault. The placement starts at the point that the local variable dest-base is defined. It places the User object in the location of the unmatched source DN, dest-base. The rule uses LDAP format.

Placement - Publisher Flat

This rule places objects from the data store into one container in the Identity Vault. Implement the rule on the Placement policy in the driver. You can implement the rule only on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 135
- "Importing the Predefined Rule" on page 135
- "How the Rule Works" on page 136

Creating a Policy

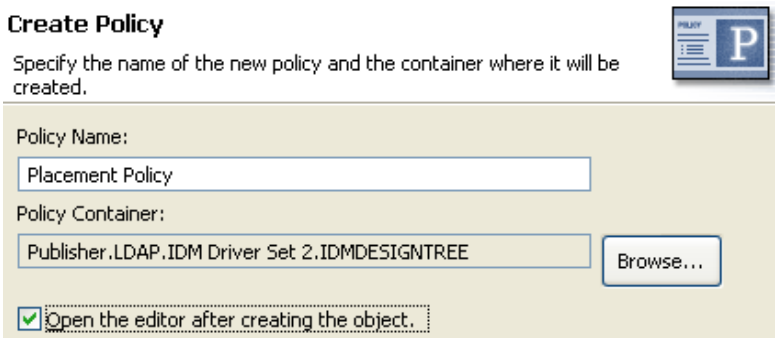
1 From the Outline view or the Policy Flow view, select the Publisher channel.

2 Select the Placement policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.

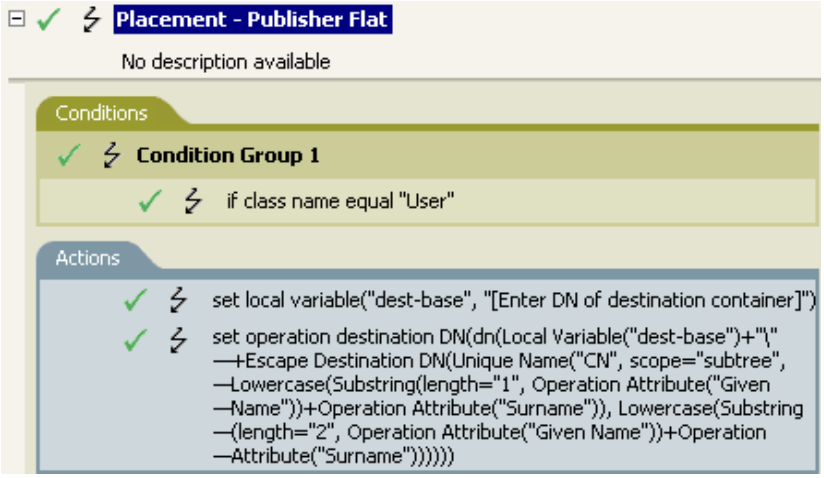
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Placement - Publisher Flat**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter DN of destination container]** from the **Specify String** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, click the browse button, then browse to and select the destination container where you want all of the User objects to be placed, then click **OK**.

10 Click **Finish**.

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule places all User objects in the destination DN. The rule sets the DN of the destination container as the local variable dest-base. The rule then sets the destination DN to be the dest-base\CN attribute. The CN attribute of the User object is the first two letters of the Given Name attribute plus the Surname attribute in lowercase. The rule uses slash format.

Placement - Subscriber Flat - LDAP Format

This rule places objects from the Identity Vault into one container in the data store. Implement the rule on the Subscriber Placement policy in the driver.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 137
- "Importing the Predefined Rule" on page 137
- "How the Rule Works" on page 138

Creating a Policy

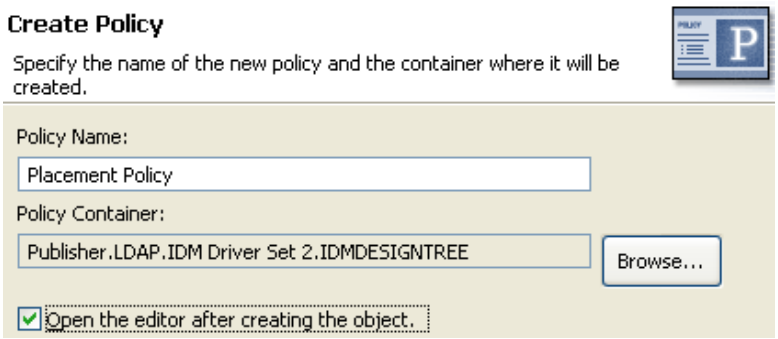
1 From the Outline view or the Policy Flow view, select the Publisher channel.

2 Select the Placement policy set in Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.

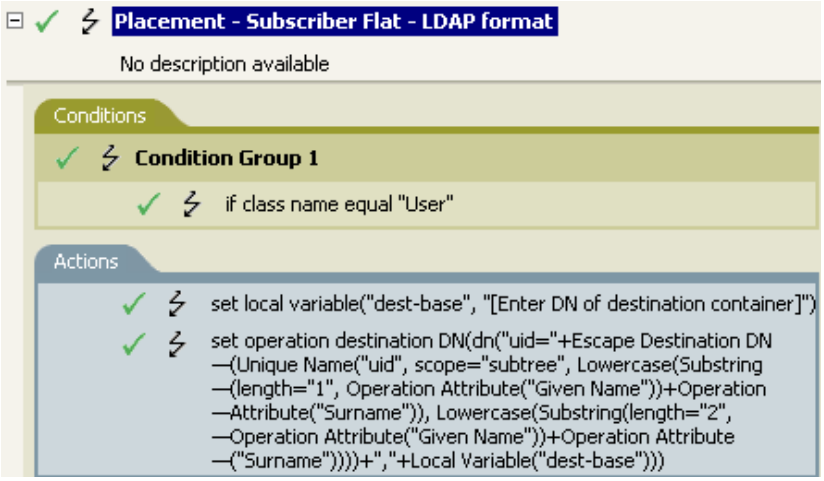
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule > Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Placement - Subscriber Flat - LDAP format**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter DN of destination container]** from the **Specify String** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, add the destination container where you want all of the User objects to be placed. Make sure the container is specified in LDAP format, then click **OK**.

10 Click **Finish**.

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule places all User objects in the destination DN. The rule sets the DN of the destination container as the local variable dest-base. The rule then sets the destination DN to be uid=unique name,dest-base. The uid attribute of the User object is the first two letters of the Given Name attribute plus the Surname attribute in lowercase. The rule uses LDAP format.

Placement - Publisher By Dept

This rule places objects from one container in the data store into multiple containers in the Identity Vault. Implement the rule on the Placement policy in the driver. You can implement the rule only on the Publisher channel.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 139
- "Importing the Predefined Rule" on page 139
- "How the Rule Works" on page 140

Creating a Policy

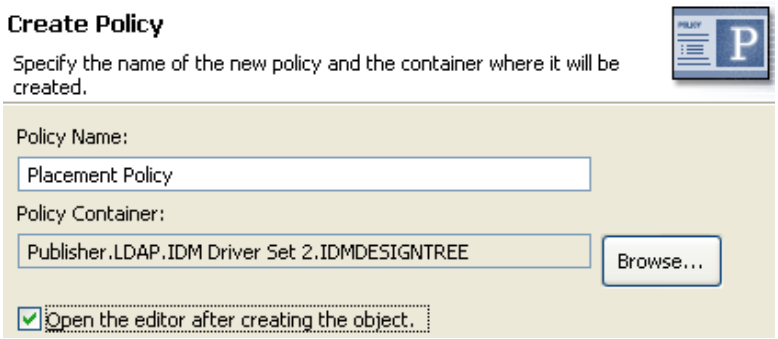
1 From the Outline view or the Policy Flow view, select the Publisher channel.

2 Select the Placement policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.

3 Click **Create a new policy**, then click **Next**.

4 Name the policy.

5 Use the default location or browse and select another location to place the policy in the driver.



6 Select **Open Editor after creating policy**, then click **Next**.

7 Select **DirXML Script** for the type of policy, then click **Finish**.

8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.

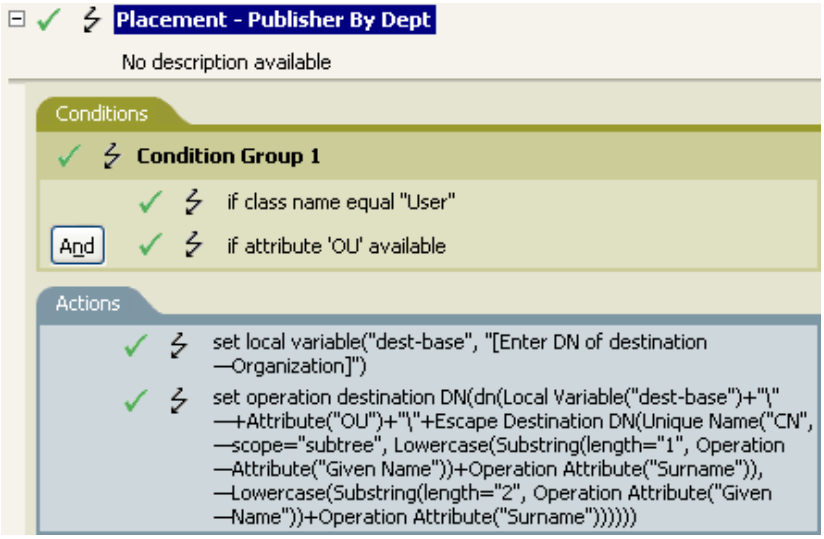
9 Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1 Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.

2 Select **Placement - Publisher By Dept**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter DN of destination Organization]** from the **Specify String** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, click the browse button, then browse to and select the parent container in the Identity Vault. Make sure all of the department containers are child containers of this DN, then click **OK**.

10 Click **Finish.**

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule places User objects in proper department containers depending upon the value that is stored in the OU attribute. If a User object needs to be placed and has the OU attribute available, then the User object is placed in the dest-base\value of OU attribute\CN attribute.

The dest-base is a local variable. The DN must be the relative root path of the department containers. It can be an organization or an organizational unit. The value stored in the OU attribute must be the name of a child container of the dest-base local variable.

The child containers must be associated for the user objects to be placed. The value of the OU attribute must be the name of the child container. If the OU attribute is not present, this rule is not executed.

The CN attribute of the User object is the first two letters of the Given Name attribute plus the Surname attribute in lowercase. The rule uses slash format.

Placement - Subscriber By Dept - LDAP Format

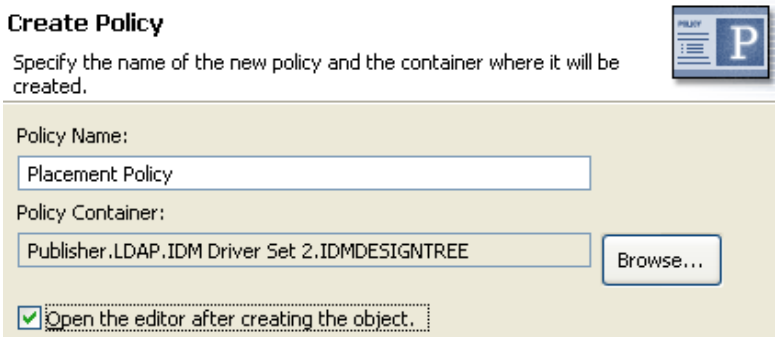
This rule places objects from one container in the Identity Vault into multiple containers in the data store based on the OU attribute. Implement the rule on the Placement policy in the driver. You can implement the rule only on the Subscriber channel.

There are two steps involved in using the predefined rules: creating a policy in the Placement policy set and importing the predefined rule. If you already have a Placement policy that you want to add this rule to, skip to Importing the Predefined Rule.

- "Creating a Policy" on page 141
- "Importing the Predefined Rule" on page 141
- "How the Rule Works" on page 142

Creating a Policy

1. From the Outline view or the Policy Flow view, select the Subscriber channel.
2. Select the Placement policy set in the Policy Set view, then click **Create or add a new policy to the Policy Set** icon to create a new policy.
3. Click **Create a new policy**, then click **Next**.
4. Name the policy.
5. Use the default location or browse and select another location to place the policy in the driver.

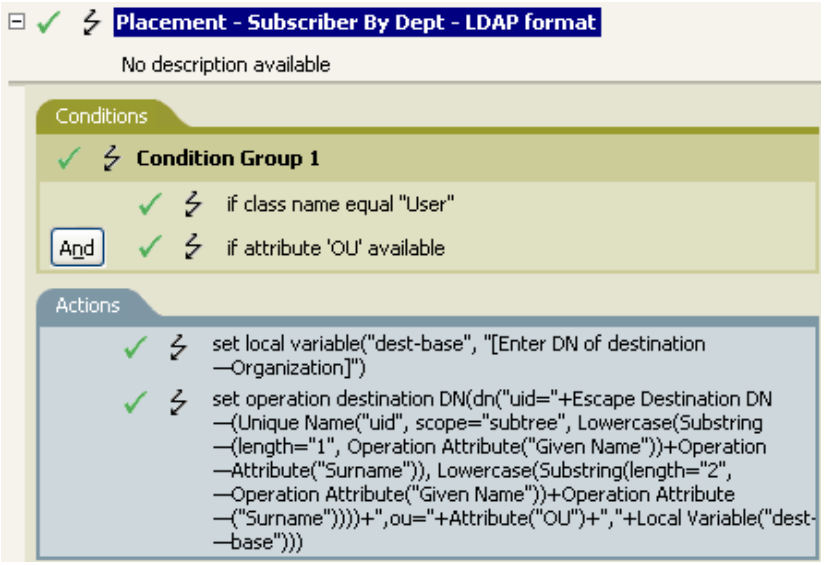


6. Select **Open Editor after creating policy**, then click **Next**.
7. Select **DirXML Script** for the type of policy, then click **Finish**.
8. A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click **Yes**. The Policy Builder is launched and the new Placement policy is saved.
9. Continue with Importing the Predefined Rule.

Importing the Predefined Rule

1. Right-click in the Policy Builder, then click **New** > **Predefined Rule** > **Insert Predefined Rule Before** or **Insert Predefined Rule After**.
2. Select **Placement - Subscriber By Dept - LDAP format**, then click **OK**.

3 Expand the predefined rule.



4 Edit the action by double-clicking the **Actions** tab.

5 Delete **[Enter DN of destination Organization]** from the **Specify string** field.

6 Click the **Edit the arguments** icon to launch the Argument Builder.

7 Select **Text** in the noun list.

8 Double-click **Text** to add it to the argument.

9 In the editor, add the parent container in the data store. The parent container must be specified in LDAP format. Make sure all of the department containers are child containers of this DN, then click **OK**.

10 Click **Finish**.

11 Save the rule by clicking **File** > **Save**.

How the Rule Works

This rule places User objects in proper department containers depending upon the value that is stored in the OU attribute. If a User object needs to be placed and has the OU attribute available, then the User object is placed in the uid=unique name,ou=value of OU attribute,dest-base.

The dest-base is a local variable. The DN must be the relative root path of the department containers. It can be an organization or an organizational unit. The value stored in the OU attribute must be the name of a child container of the dest-base local variable.

The child containers must be associated for the User objects to be placed. The value of the OU attribute must be the name of the child container. If the OU attribute is not present, then this rule is not executed.

The uid attribute of the User object is the first two letters of the Given Name attribute plus the Surname attribute as lowercase. The rule uses LDAP format.

8 Testing Policies with the Policy Simulator

The Policy Simulator allows you to test and debug a single policy or a group of policies contained in a policy set without implementing the policy in the Identity Vault. It also provides a graphical editor to create XDS Input documents. With these features, you can test the policies without affecting the production environment or the connected system.

For more information about common tasks with the Policy Simulator, see the following sections:

- "Accessing the Policy Simulator" on page 143
- "Creating an XDS Input Document" on page 144
- "Using the Operation Data Editor" on page 152
- "Using the Hex Editor" on page 152
- "Simulating a Policy" on page 160
- "Simulating Policies with Java Extensions" on page 164
- "Simulating Policies with Referenced Directories" on page 165

The Policy Simulator uses XML. The OpenText eDirectory document type definition file (`nds.dtd`) defines the schema of the XML documents that the OpenText Identity Manager engine can process. XML documents that do not conform to this schema generate errors. To verify whether the document conforms to the `nds.dtd` and to find information about why errors are occurring, see the NDS DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

If the policy uses a mapping table object or ECMAScript object, the Policy Simulator tests these objects when the policy is tested. It also allows you to test included policies and referenced GCVs.

The Policy Simulator cannot simulate the initial policy sets from application drivers such as SOAP and Delimited text. These drivers use comma-separated files or text files as input, and the XML or XDS is derived from policies in the policy chain. Currently, the Policy Simulator only accepts valid XML or XDS as input. Additional functionality is being considered for future releases.

Accessing the Policy Simulator

The Policy Simulator can be accessed in three different ways:

- "Outline View" on page 144
- "Policy Flow View" on page 144
- "Editors" on page 144

Outline View

1 Click the **Show Model Outline** icon .

2 Right-click the driver, publisher, subscriber, mapping rule, filter, or any policy you want to simulate, then select **Simulate** .

Policy Flow View

1 Click the **Show Policy Flow** icon .

2 Right-click the Input, Output, Schema Map, filter, or any policy set icons you want to simulate, then select **Simulate** .

Editors

You can access the Policy Simulator through the Policy Builder, the Schema Map editor, or the Filter editor by selecting the **Policy Simulator** icon in the toolbar of each editor.

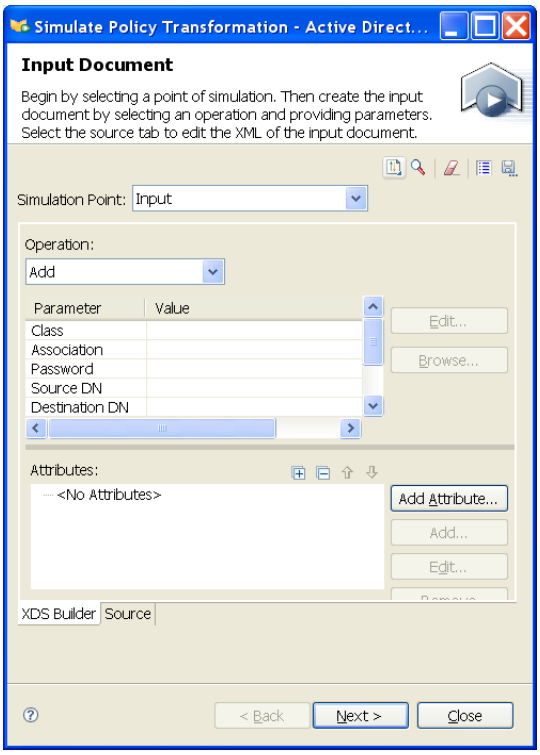
Creating an XDS Input Document

In order to simulate a policy, you must have a valid XDS Input document. The policy consumes the input document and the results are displayed as if the policy was executed. The simulator provides a graphical editor, to help you create the input document. The editor is called the XDS Builder.

You access the XDS Builder by clicking the **XDS Builder** tab in the Policy Simulator.

For information on how to access the Policy Simulator, see "Accessing the Policy Simulator" on page 143.

Figure 8-1 XDS Builder



Click the **Source** tab in the Policy Simulator to display the input document in XML. The XDS Builder creates this input document. You can modify the XML by editing the XML directly or using the XDS Builder.

The XDS Builder allows you to select the operation type as well as provide the operation parameters, attributes, and values. XDS Builder saves the parameters and values of the simulator for the current Designer session. To make the simulator input available after Designer has been shut down, save the input document to disk.

After you have created the XDS input document, you need to analyze the results. For more information, see "Simulating a Policy" on page 160.

The Policy Simulator has several different components. Each component helps create the input document to test the policy against.

- "Source" on page 146
- "Import an XDS Document" on page 146
- "Use an Identity Vault Object As a Template" on page 146
- "Clear All Parameters" on page 147
- "Configuration Options" on page 147
- "Save the Input Document" on page 147
- "Simulation Point" on page 147
- "Operation" on page 148
- "Parameter and Value" on page 148
- "Attributes" on page 149

Source

The Policy Simulator allows you to create the input document in XML without using the builder. The **Source** tab is an XML editor.

Import an XDS Document

The Policy Simulator allows you to import an existing input document to test the policy against.

1. In the toolbar, select **Import an XDS input document from a file** .
2. Browse to and select the existing input document, then click **Open**.
3. Click **Next** to test the policy against this existing information.

Designer comes with sample input document files you can use. The files are located in the plug-in `com.novell.designer.idm.policy\simulation`. The events are Add, Association, Delete, Instance, Modify, Move, Query, Rename, and Status.

Use an Identity Vault Object As a Template

The Policy Simulator allows you to use an existing Identity Vault object to populate the input document.

1. In the toolbar, select **Browse to an object in the Identity Vault to use as a template** .
2. If you are not logged in to the Identity Vault, specify the following information; otherwise skip to Step 3.
   - **2a** Specify the host name of the Identity Vault server.

     It can be the IP address of the server or the DNS name of the server.
   - **2b** Specify a DN of a user object to authentication to the Identity Vault.
   - **2c** Specify the password of the user in Step 2b, then click **OK**.
3. Browse to and select the desired object, then click **OK**.

   If the simulation point is set to **Input**, **Output**, or **Schema Map Inbound**, a warning message is displayed. (For more information about simulation points, see "Simulation Point" on page 147.) The warning message informs the user that the input document should be created by using the application's attribute names and value formats. The XDS Builder converts the Identity Vault attribute names to the corresponding application attribute by using the Schema Map policy, as long as the driver references the Schema Map policy. However, the values for the attributes might be in an incorrect format.
4. Click **OK** if a warning message is displayed.
5. Click **Next** to test the policy against the object.

Clear All Parameters

The Policy Simulator allows you to clear all parameters, attributes, and values that have been set. You can create a new input document without launching the XDS Builder again.

1. In the toolbar, select **Clear** .
2. Specify information to create a new input document.

Configuration Options

The Policy Simulator allows you to set configuration options for the simulation.

1. In the toolbar, select **Configure options for the simulation** .
2. Specify the desired **XSL Trace Level**.

   If you have XSL in your policy and you want to see the XSL trace results, specify a value. If the value is set to 0, no information is displayed. The range of the trace value is 0 to 4.
3. Specify the desired **Driver Trace Level**.

   To set the results of the simulation, set a value in the **Driver Trace Level** field. The range of the trace value is 0 to 5.
4. Click **OK**.

Save the Input Document

The Policy Simulator does not store the input document for future sessions in Designer. If you want to use the input document for a later session in Designer, the input document must be saved.

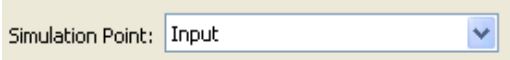
1. In the toolbar, click **Save** .
2. Browse to a location where you want to save the file, then specify a filename.
3. Click **Save** to save the input document.

We recommend that you do not save the input document in the same directory where Designer is installed or it might be overwritten during a Designer upgrade.

Simulation Point

The Policy Simulator allows you to select a policy set or group of policies to simulate with a specific operation.

Figure 8-2 *Simulation Point in the Policy Simulator*



You can select a Driver object, Publisher channel, Subscriber channel, policy, or rule as the simulation point. If you select a Driver object, Publisher channel, or Subscriber channel, the **Simulation Point** options are:

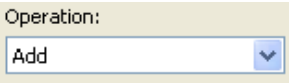
Input	**Publisher Placement**	**Subscriber Command**
Schema Map Inbound	**Publisher Command**	**Subscriber Placement**
Publisher Event	**Publisher Notify Filter**	**Subscriber Creation**
Publisher Sync Filter	**Output**	**Subscriber Matching**
Publisher Matching	**Schema Mapping Outbound**	**Subscriber Event**
Publisher Creation	**Subscriber Notify Filter**	**Subscriber Sync Filter**

NOTE: If you want to test a single policy, launch the simulator from the selected policy. If you do select a specific policy or rule to test, the **Simulation Point** options are **To Identity Vault** and **From Identity Vault**.

Operation

The XDS Builder allows you to select the type of operation that the input document performs.

Figure 8-3 *Operation Options in the XDS Builder*



The available operations are:

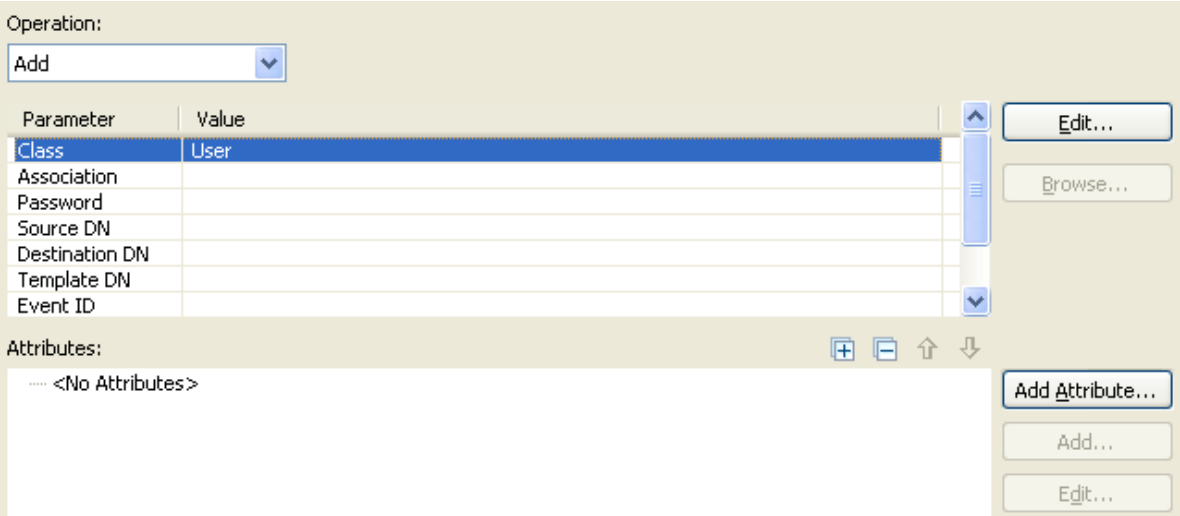
Add	Modify	Remove Association
Add Association	Modify Association	Rename
Check Object Password	Modify Password	Status
Check Password	Move	Sync
Delete	Query	Instance
Get Named Password	Extended Query	

Depending upon which operation is selected, the XDS Builder displays different options and screens.

Parameter and Value

The XDS Builder allows you to define parameters and specify values for the selected operation. Each operation displays different parameters.

Figure 8-4 *Parameters and Values in the XDS Builder*



The list of parameters for each operation is set, and cannot be changed. You do not need to have each parameter defined for the simulation to work; just define the parameters that apply to your policy. You can edit the parameter value by double-clicking the value or selecting the value and clicking the **Edit** button.

All parameter values are edited inline, with the exception of **Class** and **Operation Data** parameters. Editing these parameters launches a dialog box that allows you to select a class name or edit the operation data.

Parameters that contain a reference to an object enable the **Browse** button. Although these values can be edited inline, the **Browse** button allows you to browse for an object in the Identity Vault.

Editing the **Class** parameter launches the application class browser when the Input, Output, or Schema Map inbound policy simulation point is selected. For all other simulation points, the Identity Vault class browser is opened. If the desired class is not included in the application or Identity Vault schema, it can be added during the simulation process. For more information about managing a schema in Designer, see "Class List Toolbar" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

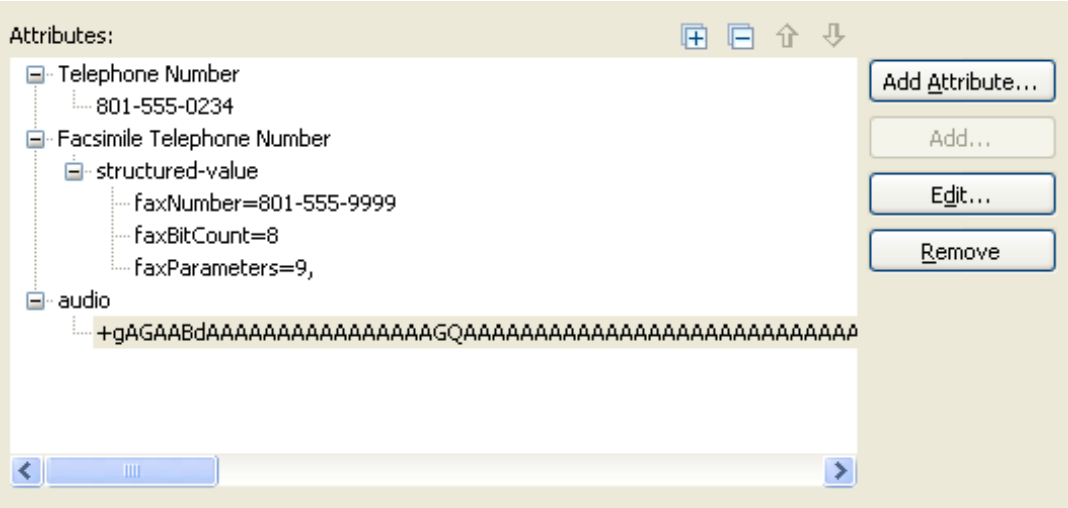
Editing the **Operation Data** parameter launches the Operation Data editor. For more information, see "Using the Operation Data Editor" on page 152.

Attributes

The Input Document **Attributes** field allows you to add, edit, and remove attribute values for simulating operations.

- "Working with Attribute Values" on page 150
- "Simulating the Adding of an Attribute" on page 151
- "Simulating the Modification of an Attribute" on page 151

Figure 8-5 XSD Input Document - Attribute Field



Working with Attribute Values

Because there are several different attribute types, the Attributes field provides different ways of manipulating attribute values.

- **Add a New Attribute:** To add a new attribute to the attribute list, click **Add Attribute**. For more information, see "Simulating the Adding of an Attribute" on page 151.
- **Remove an Attribute:** To remove an attribute from the attribute list, select the attribute, then click **Remove** (or right-click the attribute, then select **Remove**.)
- **Add an Attribute Value:** To add another value to an existing attribute, select the attribute, then click **Add** (or right-click the attribute, then select **Add**.
- **Remove an Attribute Value:** To remove a value from an existing attribute, select the value, then click **Remove** (or right-click the value, then select **Remove**.)
- **Change an Attribute Value:** To change the value of an existing attribute, select an attribute value, then click **Edit** (or double-click an attribute value.)

  If the attribute uses a structured value, you can change each of the value components separately. You cannot modify the entire structured value at once.

  If the attribute is an octet string, Simulator opens the Hex Editor to modify the value. For more information, see "Using the Hex Editor" on page 152.
- **Identity Vault Schema** When working with Identity Vault attributes with structured values, the Simulator displays customized Value editor dialog boxes that describe each of the structured value components. For example, adding a Facsimile Telephone Number attribute launches a Value Editor dialog box that asks for the Fax Number, Bit Count, and Parameters for the attribute, each of which is a component of the Facsimile Telephone Number structured value.

  However, when working with an Application schema, the Simulator uses a generic structured value dialog box since it cannot know the type of data that comprises the structured value.

Simulating the Adding of an Attribute

1. Select **Add** in the **Operation** field of the Simulator.
2. Double-click **Class** in the **Parameter** field of the Simulator.
3. Browse to and select the desired class, the click **OK**.
4. Click **Add Attribute**.

   The **Add Attribute** icon launches the Identity Vault or application attribute browser, based on the simulation point.
5. Browse to and select the desired attribute, then click **OK**.
6. Specify the attribute value, then click **OK**.

   Based on the attribute type, Simulator opens either the Value editor or the Hex editor so you can specify the attribute value.
7. Click **Next** in the Simulator to view the results of the Add operation with the specified attribute value.

Simulating the Modification of an Attribute

There are multiple events that cause an attribute to be modified. They are:

- **Add Value:** Adds a new value to the attribute.
- **Remove Value:** Removes a single value from the attribute.
- **Remove All Values:** Removes all values stored in the attribute.
- **Remove:** Removes the attribute.

When you are simulating a Modify operation, you need to select which event occurs to modify the attribute. The Simulator allows you to do that:

1. In the Policy Simulator, select **Modify** in the **Operation** field of the XDS Builder.
2. Double-click **Class** in the **Parameter** column.
3. Browse to and select the desired class, then click **OK**.
4. Click the **Add Attribute** button.
5. Browse to and select the desired attribute, then click **OK**.
6. Right-click the attribute, then select one of the modifying events:
   - Add Value
   - Remove Value
   - Remove All Values
   - Remove

   You can add multiple events to a single attribute.
7. Click **Next** in the Policy Simulator to view the results of the Modify operation.

The Policy Simulator allows you to modify the values of the attribute and change the order of events that occur to an attribute. When you right-click an event in the **Attributes** field, you have additional options that allow to make these changes:

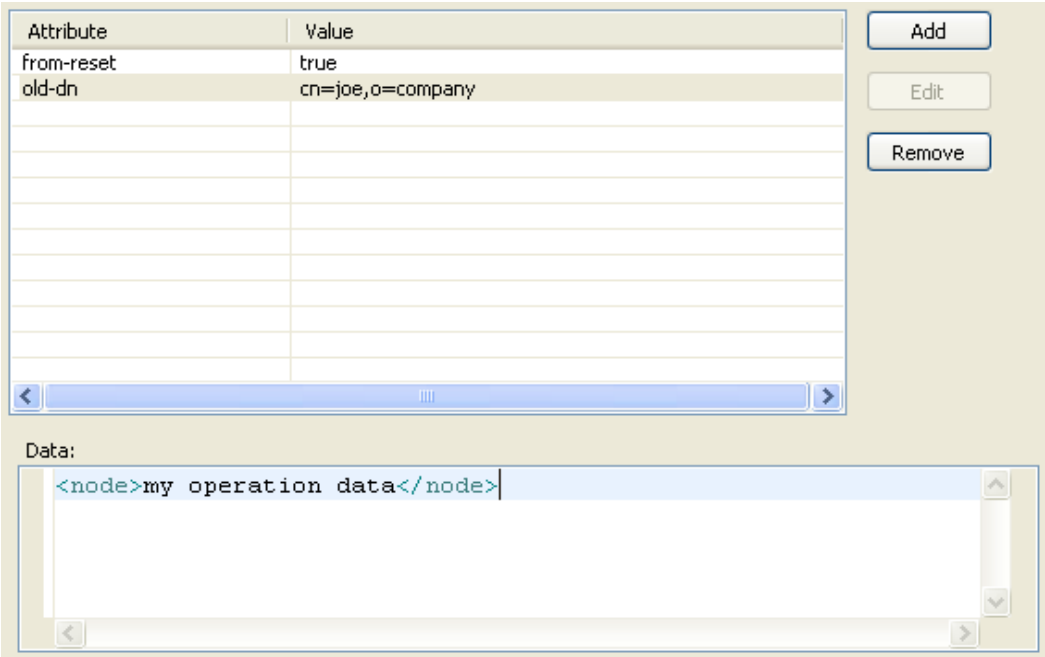
- **Add:** Allows you to add content to the attribute value.

- **Change to Add Value/Change to Remove Value:** Allows you to change the event from Add Value to Remove Value or vice versa.
- **Remove:** Removes the selected event from the list of events to occur on an attribute.
- **Move Up:** Moves the selected event up in the order of execution.
- **Move Down:** Moves the selected event down in the order of execution.

Using the Operation Data Editor

The Operation Data editor allows you to create an operation data element for the selected operation by specifying attributes and values that should be included in the node. An XML fragment should also be included in the node.

1. In the **Parameter** column of the Policy Simulator, double-click the **Operation Data** field.
2. In the Operation Data editor, click **Add** to add the desired attribute.
3. Specify the name of the attribute in the **Attribute** field.
4. Specify the value of the attribute in the **Value** field.
5. If you want to add an additional attribute, repeat Step 2 through Step 4.
6. Click the **Data** field, then specify the XML fragment.

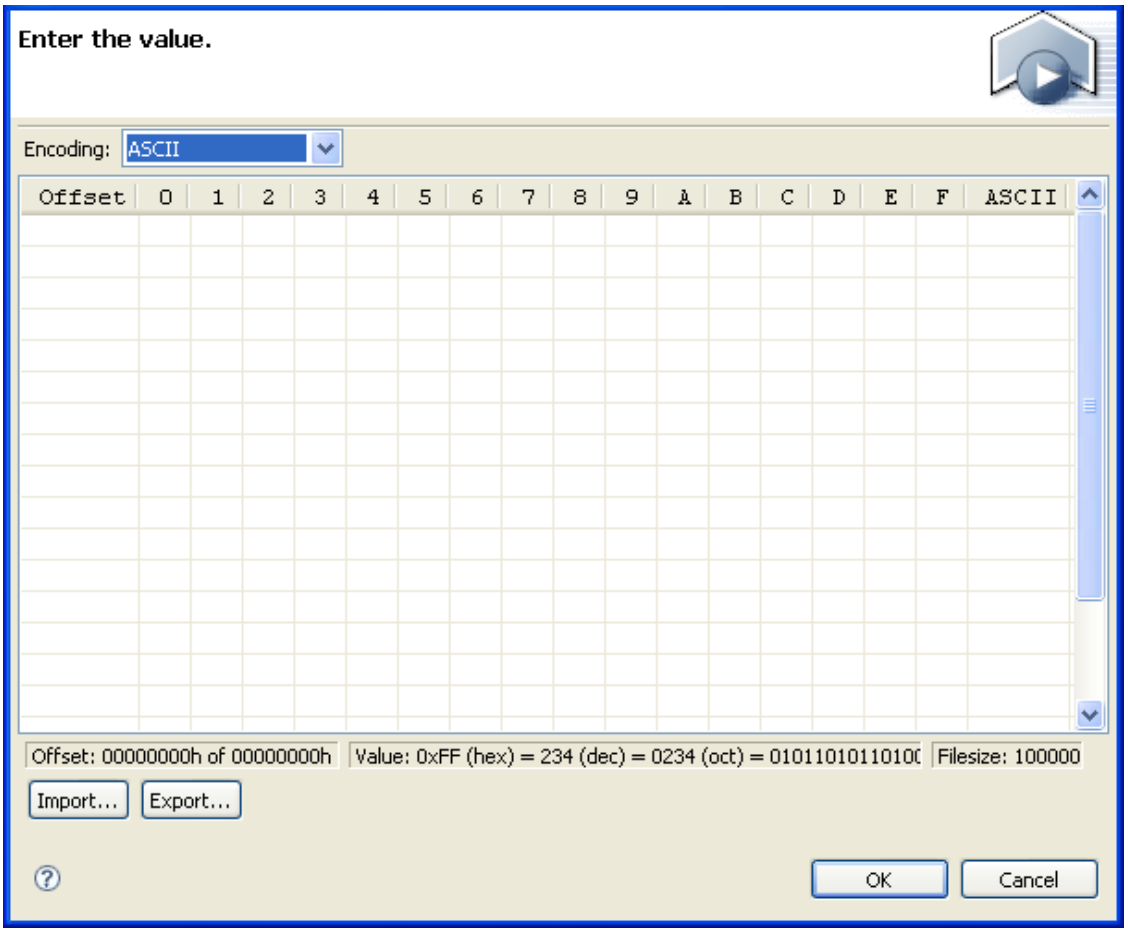


7. Click **OK** to save the information.

Using the Hex Editor

The Hex editor allows you to view or edit any attribute values in hex mode. For example, if you are synchronizing OpenText eDirectory attribute values of type octet string, then you can edit this information through Designer.

Figure 8-6 *Hex Editor*



- “Accessing the Hex Editor” on page 154
- “Importing Data into the Hex Editor” on page 154
- “Inserting Data in the Hex Editor” on page 155
- “Appending Data in the Hex Editor” on page 155
- “Editing Data in the Hex Editor” on page 156
- “Reverting Changes in the Hex Editor” on page 158
- “Deleting Data in the Hex Editor” on page 158
- “Moving the Cursor in the Hex Editor” on page 159
- “Exporting Data from the Hex Editor” on page 159

Accessing the Hex Editor

The Hex editor is inside of the Policy Simulator. The Hex editor is opened for all attributes that have an OpenText eDirectory syntax of octet string or unknown and an application syntax type of octet. You can also access the Hex editor by following these steps:

1 Launch the Policy Simulator and do the following:

   1a Set the **Simulation Point** to **Publisher Creation**.

   1b Add a class parameter of **User**.

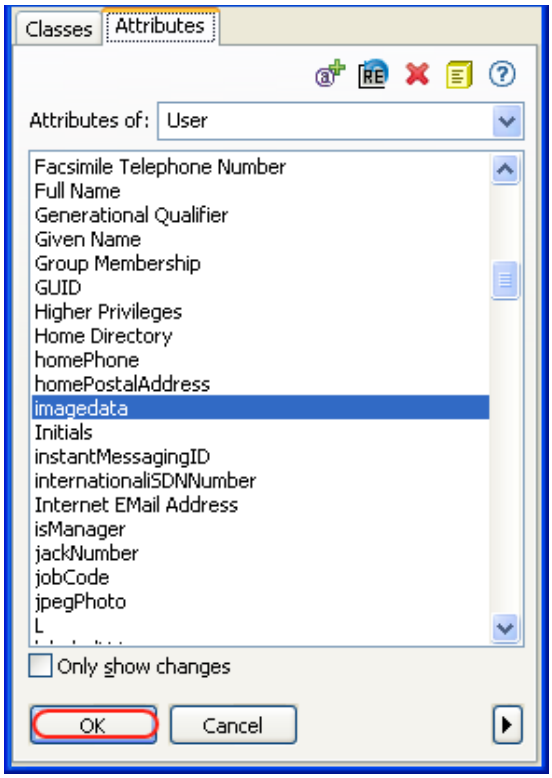
   1c Click the **Add Attribute** button to add a new attribute to the class.

2 In the Schema Browser, select **Add an Attribute** .

   Follow the steps in the New Attribute Wizard to create a new attribute. Make sure you specify the attribute's syntax type as **Octet String**.

   For more information, see "Creating Identity Vault Attributes" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

3 In the Schema Browser, select the new attribute, then click **OK** to launch the Hex editor.



Importing Data into the Hex Editor

You can import data from a file into the Hex editor.

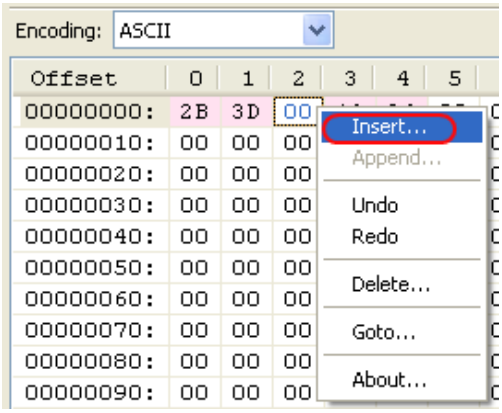
1 Click **Import** in the Hex editor.

2 Browse to and select the file that has the information to import, then click **Open**.

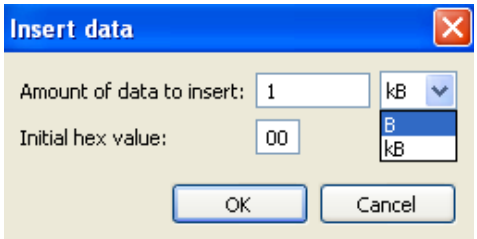
Inserting Data in the Hex Editor

You can press the Insert key to insert a single byte, or you can use the following method to add multiple bytes:

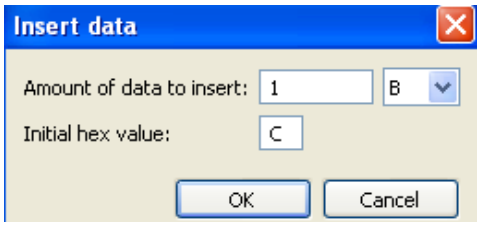
1 Select where you want to insert new data, then right-click in the Hex editor and select **Insert**.



2 Specify the amount of data to add in bytes (**B**) or kilobytes (**kB**).

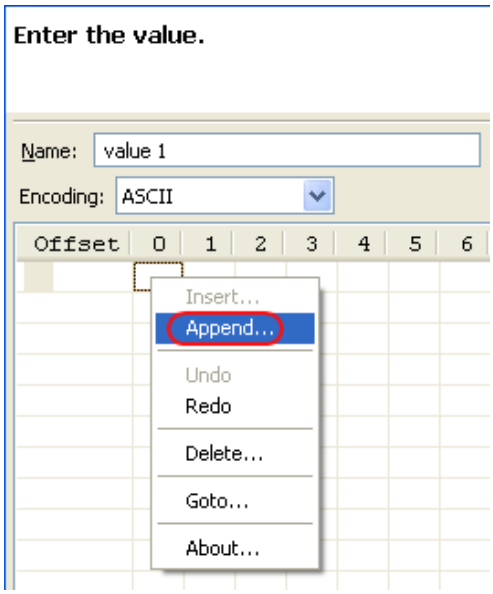


3 Specify the initial hex value, then click **OK**.



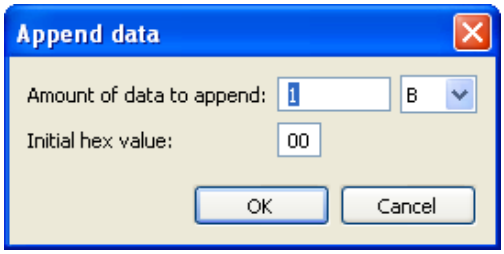
Appending Data in the Hex Editor

1 Right-click in the Hex editor, then select **Append**.



The **Append** option is available when you right-click the first byte in the table, if there is no data. It is also available when you right-click the last byte if there is data.

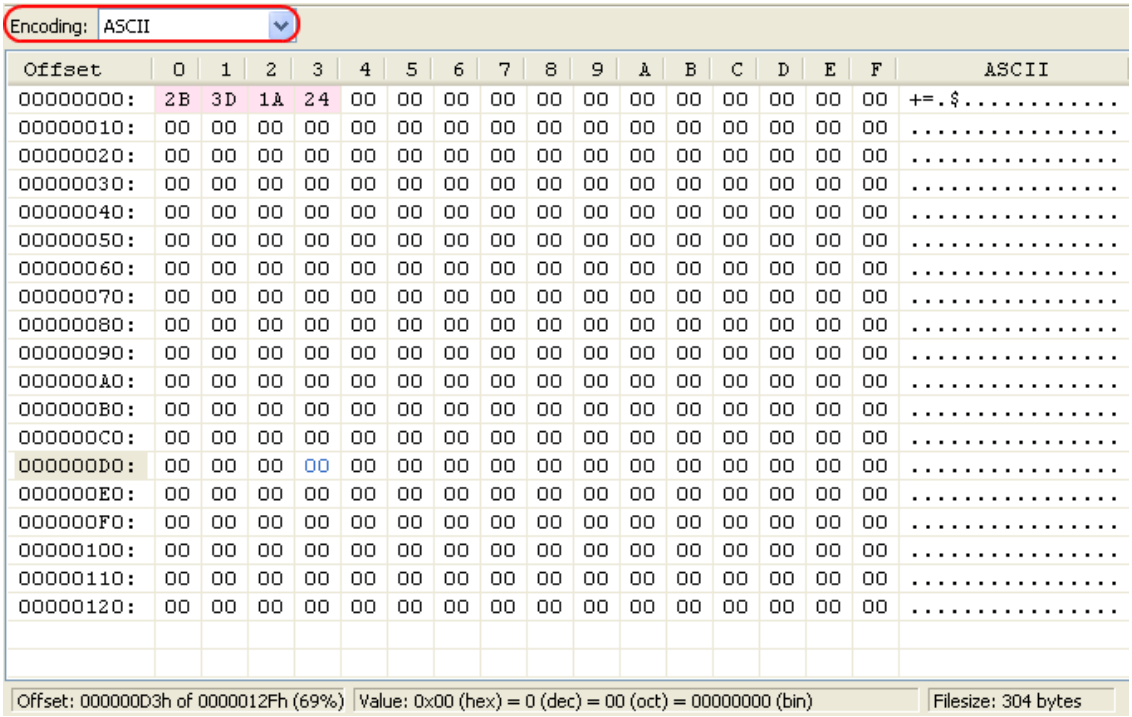
2 Specify the amount of data to append in bytes or kilobytes.



3 Specify the initial hex value, then click **OK**.

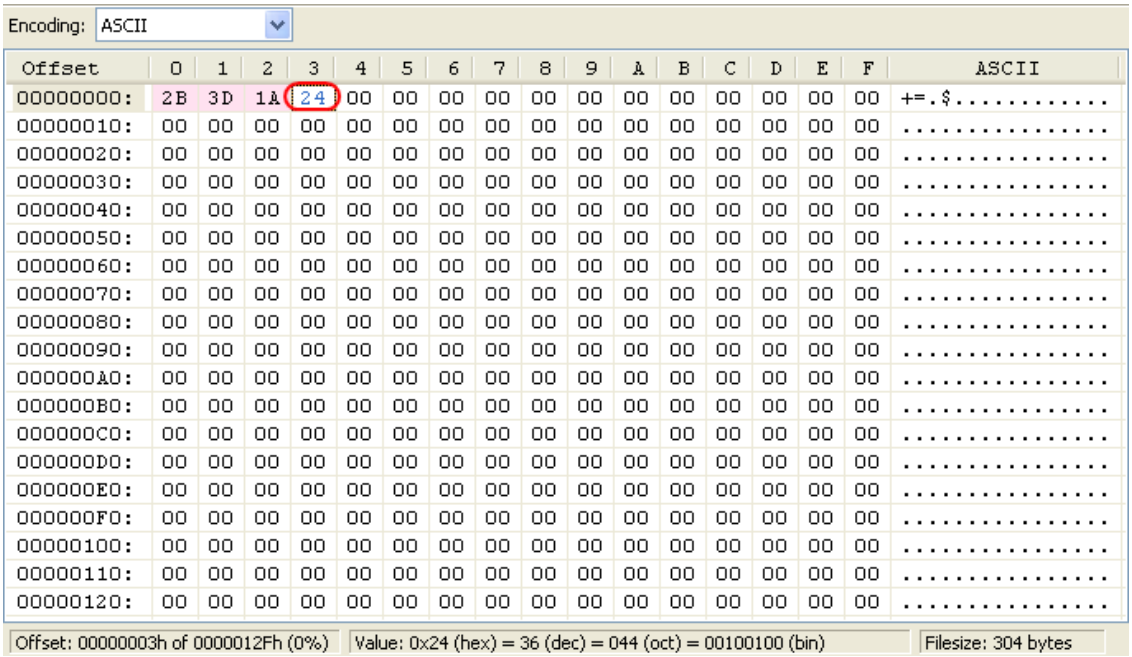
Editing Data in the Hex Editor

1 From the **Encoding** drop-down list, select the desired encoding for the value.



When the encoding is selected, the far right column displays the value encoded.

2 Select the cell of data to edit, then edit the data.



When a cell is selected, the value is displayed in blue.

3 Click **OK** to save the changes.

The Hex editor also displays the value as hex, decimal, octet, and binary.

Figure 8-7 Value Displayed in Multiple Formats

Encoding: ASCII

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
00000000:	2B	3D	1A	24	00	00	00	00	00	00	00	00	00	00	00	00	+=.$.....
00000010:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000020:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000090:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

Offset: 000000D3h of 0000012Fh (69%) Value: 0x00 (hex) = 0 (dec) = 00 (oct) = 00000000 (bin) Filesize: 304 bytes

Reverting Changes in the Hex Editor

If you make a change in the Hex editor and want to undo it:

1. Right-click in the Hex editor, then select **Undo**.

   The last change you had made is undone.

If you decide you want that change back:

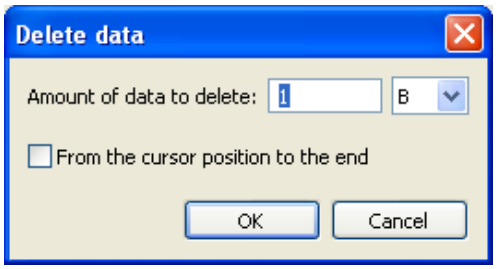
1. Right click in the Hex editor, then select **Redo**.

   The change that was undone is now redone.

Deleting Data in the Hex Editor

You can delete a single byte of data by pressing the Delete key. The Hex editor also allows you to delete sections of data from the table:

1. Right-click in the Hex editor, then select **Delete**.
2. Specify the amount of data to delete in bytes or kilobytes, then click **OK**.



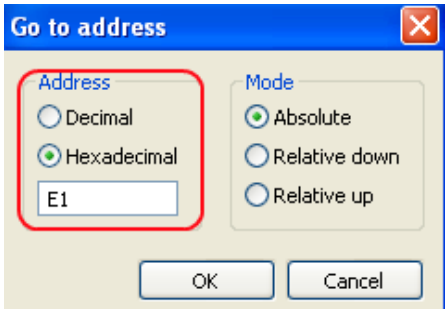
Data is deleted from the current cursor position.

If you select **From the cursor position to the end**, it deletes all data in the Hex editor from the cursor position in the table to the end of the table.

Moving the Cursor in the Hex Editor

You can move the cursor to a specified position in the Hex editor:

1. Right-click in the Hex editor, then select **Goto**.
2. Select whether the address specified in the table is a **Decimal** or **Hexadecimal** offset, then specify the value.



3. Select the mode of moving the cursor:
   - **Absolute:** Moves the cursor to the specified offset.
   - **Relative Down:** Moves the cursor down from where the cursor is currently located in the Hex editor.
   - **Relative Up:** Moves the cursor up from where the cursor is currently located in the Hex editor.
4. Click **OK** to move the cursor.

Exporting Data from the Hex Editor

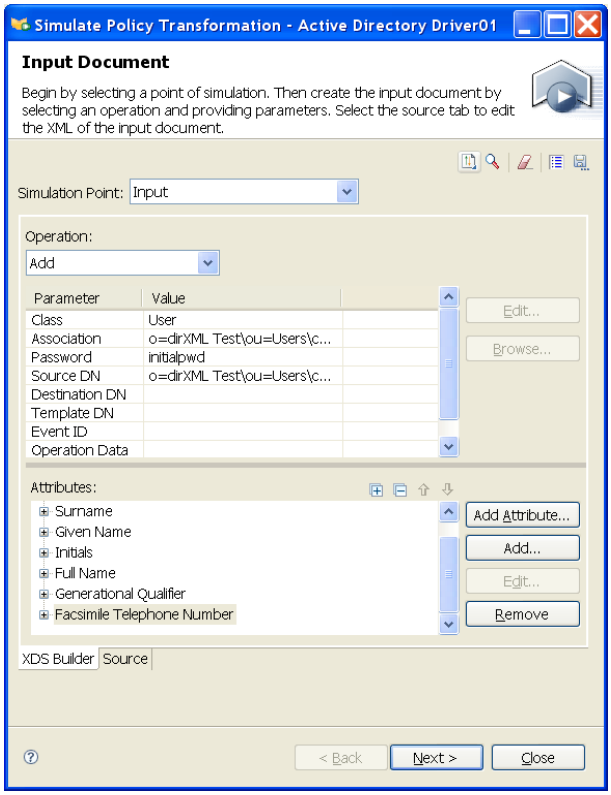
You can export data from the Hex editor to a file.

1. Click **Export** in the Hex editor.
2. Specify a filename and location for the file, then click **Save**.

Simulating a Policy

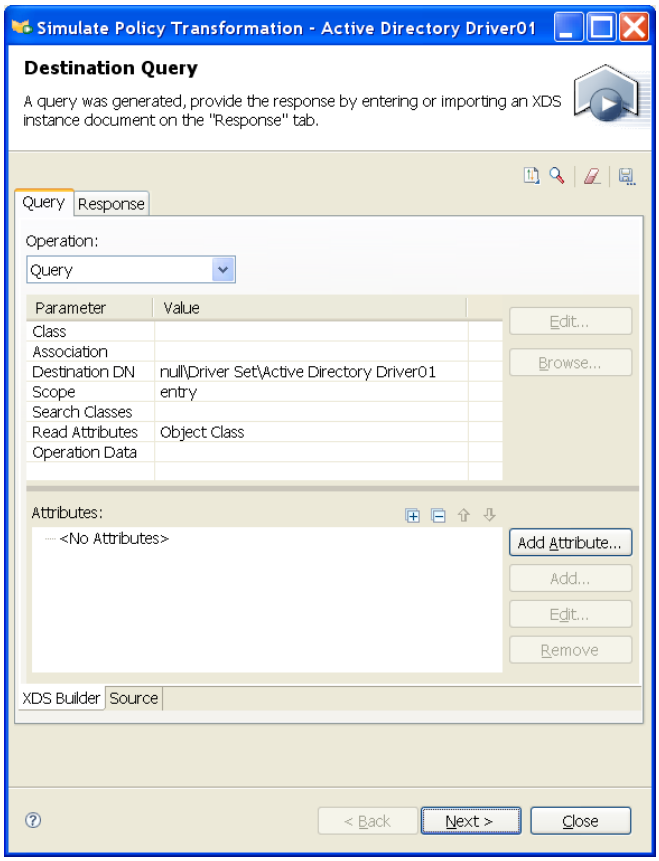
After the XDS input document has been created, you can use it to simulate the behavior of a policy.

1 In the Policy Simulator, after the XDS input document is complete, click **Next**.



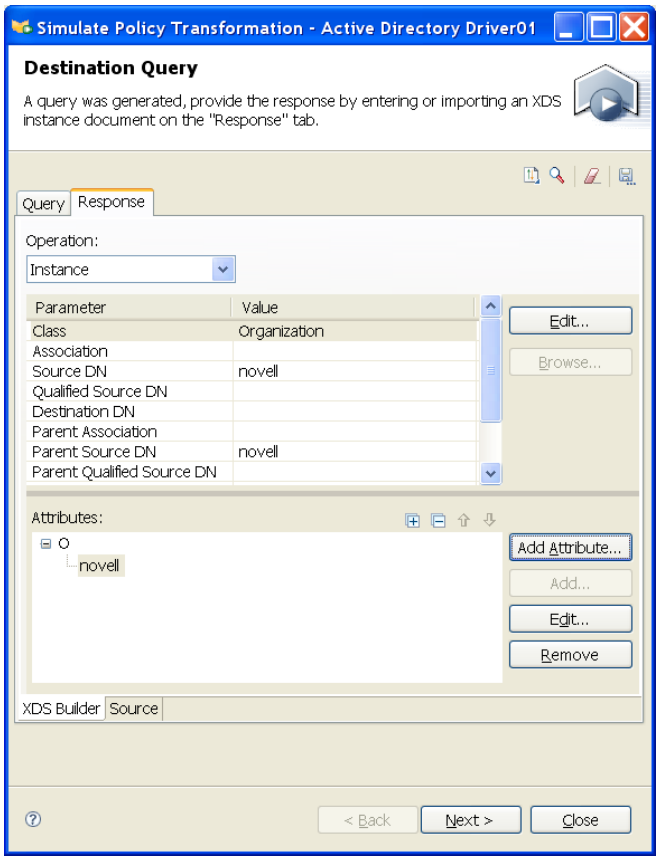
2 If the policy you are simulating generates a query, review the query in the **Query** tab, and model the query response in the **Response** tab, then click **Next**.

NOTE: As with Input Documents, you can browse the Identity Vault for objects that you want to use as templates for the simulator query or response.



Field	Description
Parameter Table	Displays the query parameters generated during the policy simulation. This matches the XML displayed in the **Source** tab. For information on using the Parameter table, see "Parameter and Value" on page 148. You can adjust the query parameters to vary the response generated when you send the query to the Application or ID Vault.
Attributes Field	Allows you to refine your query by searching for objects that contain particular values. For information on the Attributes field, see "Attributes" on page 149.
Submit to Vault	Sends the specified query to the Identity Vault to generate a Response instance document. The Simulator determines the query destination automatically and displays the appropriate button. **Submit to Vault** requires valid associations in the Association parameter. This is typically possible only when the ID Vault is deployed.

Similarly, you can model the Response instance page from the Response tab, which displays a list of objects that satisfy the query. The policy uses this response data to determine what it should do.

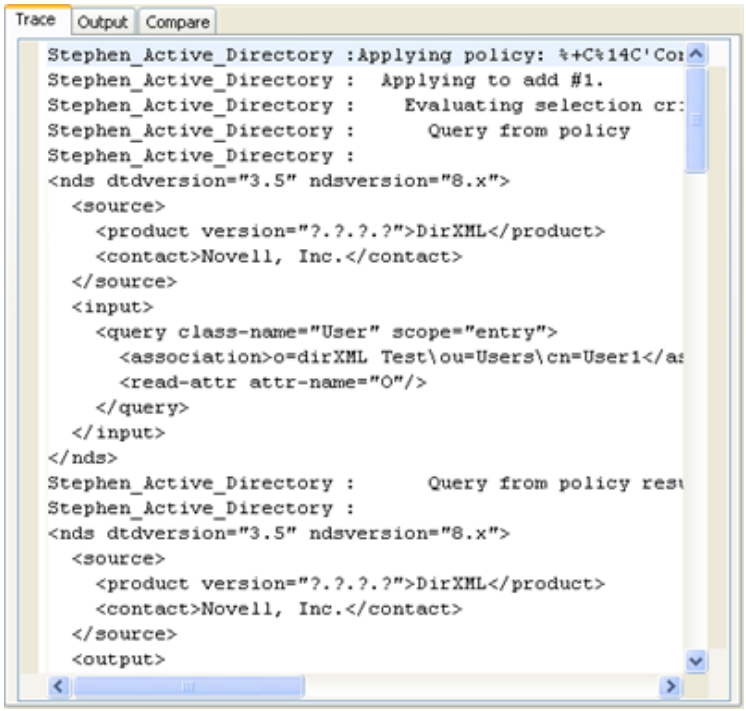


Field	Description
Parameter Table	Displays the parameters of the response instance document. This matches the XML displayed in the **Source** tab. For information on using the Parameter table, see "Parameter and Value" on page 148.
Attributes Field	Allows you to modify the response by adding or modifying the attributes in the instance document. For information on the Attributes field, see "Attributes" on page 149.

3 Click **Next**.

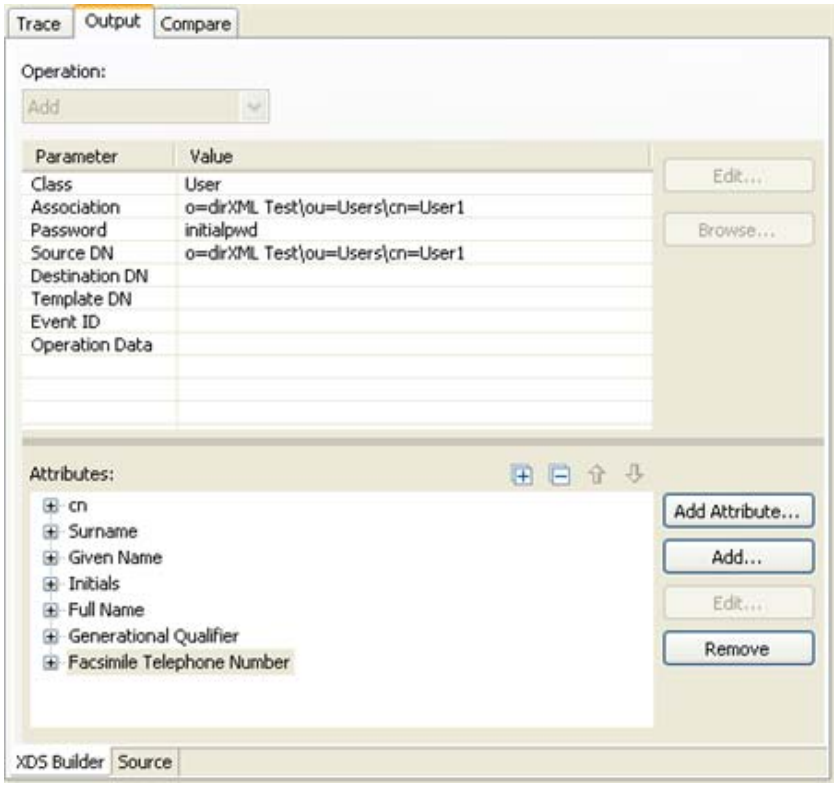
4 In the View Transform Results page, examine the results of the transformation based on your defined XDS input document.

Trace: The **Trace** tab displays the events that occur when the policy processes the defined input document. This is similar to what you would see in DS Trace if the policy processed the same input in a live IDM environment. You can configure the level of trace detail. For more information, see "Configuration Options" on page 147.

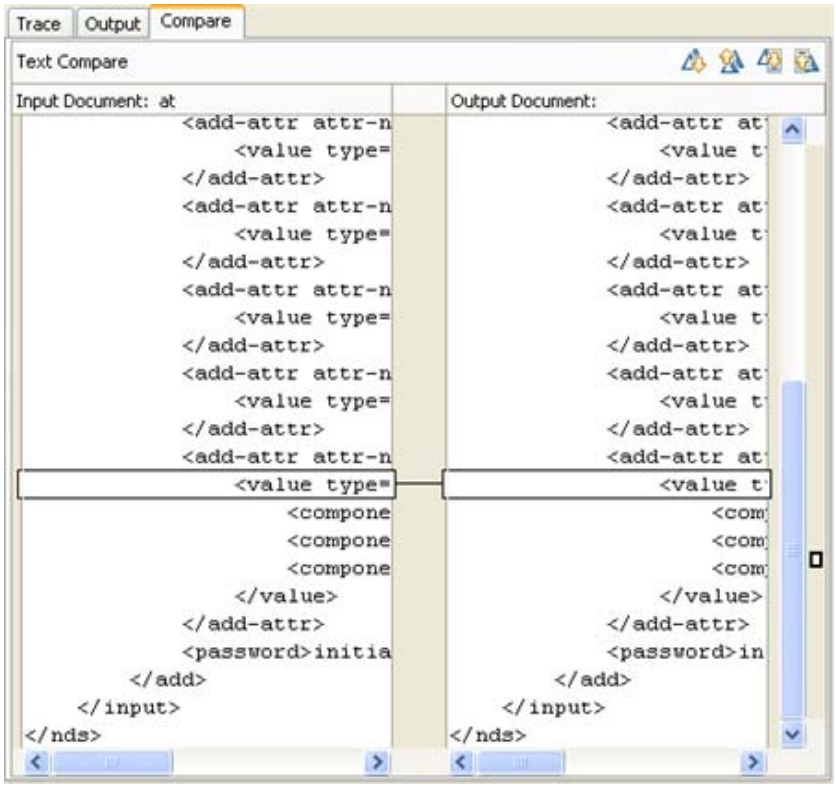


Output: The **Output** tab displays the output document generated when the policy processes the defined input document.

You can edit the output document in the XDS Builder or through the **Source** tab. If the operation was vetoed, the operation listed in the **Output** tab is indicated.



Compare: The Compare tab displays the input document and the output document side-by-side so you can examine the changes resulting from the policy processing of the input document.



5 After examining the policy effects on the input document, click one of the buttons at the bottom of the View Transform Results page:

Back: Re-opens the Input Document page so you can repeat the simulation with a different settings.

Next: Uses the current output document as the input document for the next policy set in the driver. This lets you examine how the policies work together as data flows from one policy to another.

Close: Closes the Policy Simulator.

Simulating Policies with Java Extensions

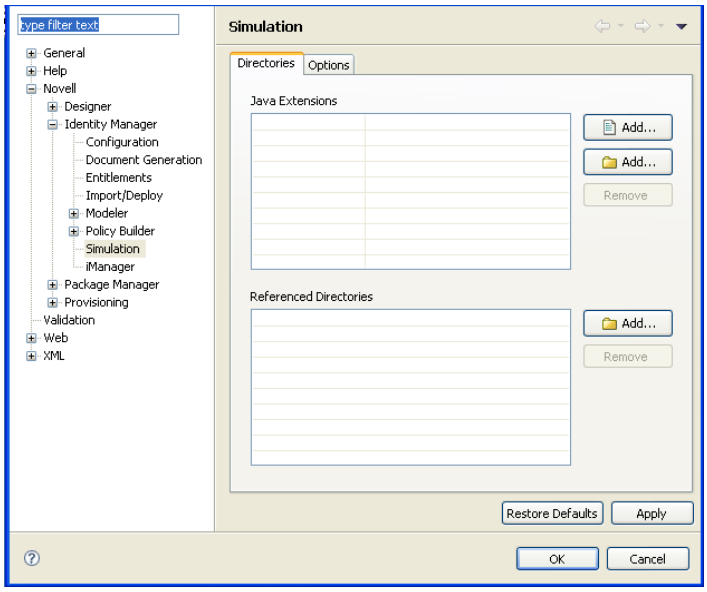
Policies that contain references to external Java extensions can now be simulated by adding the appropriate `.jar` file or directory to the class path.

To add a `.jar` file or directory to the Java class path:

1 Select **Windows > Preferences** from the tool bar.

2 Navigate to the **NetIQ > Identity Manager > Simulation** page.

3 Copy the `.jar` file containing the Java class to the specified directory and simulate the policy.



4 Click **Apply** to save your changes, or click **OK** to save your changes and close the window.

Designer allows you to specify more than one directory that contains the external Java classes.

1 Click **Add files** to select a specific `.jar` file to add to the class path. Alternatively, click **Add directories** to add all `.jar` files in the specified directory to the class path.

2 Browse to and select the desired file or directory, then click **OK**.

3 To remove a file or directory entry from the Java Extensions list, select the appropriate entry, then click **Remove**.

Simulating Policies with Referenced Directories

Policies that contain references to external directories can be simulated by adding the appropriate `.jar` file or directory to the class path.

To add a referenced directory or the Java extension:

1 Select **Windows** > **Preferences** from the tool bar.

2 Click **NetIQ** > **Identity Manager** > **Simulation**.

3 Click **Add** to add the Java extension or the referenced directory, then click **OK**.

4 Simulate your policy and the Java class path is executed.

9 Storing Information in Resource Objects

Resource objects store information that drivers use. The resource objects can hold arbitrary data in any format. OpenText Identity Manager contains different types of resource objects.

- "Generic Resource Objects" on page 167
- "Mapping Table Objects" on page 168
- "ECMAScript Objects" on page 168
- "Credential Application Objects" on page 169
- "Credential Repository Objects" on page 169
- "Package Objects" on page 169
- "Library Objects" on page 170

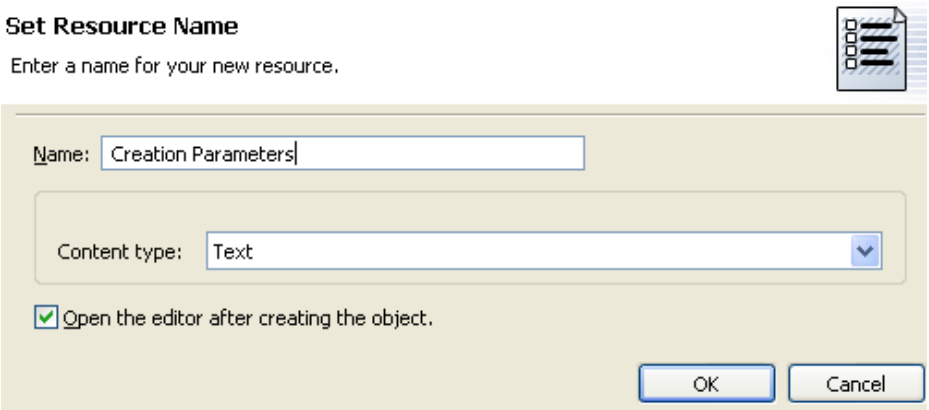
Generic Resource Objects

Generic Resource objects allow you to store information that a policy consumes. It can be any information stored in text or XML format. A resource object is stored in a library or driver object. An example of using a resource object is when multiple drivers need the same set of constant parameters. The resource object stores the parameters and the drivers use these parameters at any time.

- "Creating a Generic Resource Object" on page 167
- "Using a Generic Resource Object" on page 168

Creating a Generic Resource Object

1. In the Outline view, right-click the location where you want to create the resource object, then select **New > Resource**.
2. Specify the name of the resource object.
3. Select the content type: **XML** or **Text**.
4. Select the check box for **Open the editor after creating the object**, then click **OK**.



5 Click **Yes** in the file conflict messages.

6 Specify the desired text or XML, then press Ctrl+S to save the resource object.

Using a Generic Resource Object

A resource object is a place to store information. It is an OpenText eDirectory object, and to use the information in the object, you treat it as any other OpenText eDirectory object. The attribute DirXML-Data stores the information in the resource object, and the attribute DirXML-Content type stores the label of the information.

To read the information stored in the resource object, use the Source Attribute (page 374) or Destination Attribute (page 350) tokens. To write information to the object, use the following actions:

- Clear Destination Attribute Value (page 254)
- Clear Source Attribute Value (page 256)
- Set Default Attribute Value (page 310)
- Set Source Attribute Value (page 324)

Mapping Table Objects

A mapping table object is used by a policy to map a set of values to another set of corresponding values. After a mapping table object is created, the Map (page 392) token maps the results of the specified tokens from the values specified in the mapping table. For more information, see "Mapping Table Editor" on page 58.

ECMAScript Objects

ECMAScript objects are resource objects that store ECMAScripts. The ECMAScript is used by policies and style sheets. For more information, see Chapter 10, "Using ECMAScript in Policies," on page 173.

Credential Application Objects

Application objects store authentication parameter values for Credential Provisioning policies. There application objects for SecureLogin and for SecretStore. For information on how to create application objects for SecureLogin, see "Creating an Application Object" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. For information on how to create application objects for SecretStore, see "Creating an Application Object" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.

Credential Repository Objects

Repository objects store static configuration information for Credential Provisioning policies. There are repository objects for SecureLogin and for SecretStore. For information on how to create repository objects for SecureLogin, see "Creating a Repository Object" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. For information on how to create repository objects for SecretStore, see "Creating a Repository Object" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.

Package Objects

There are resource objects that are used to make package management work. For more information about packages, see "Understanding Packages" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

- "DS Objects" on page 169
- "Package Prompts" on page 170
- "Filters" on page 170

DS Objects

A DS Resource object contains information that creates OpenText eDirectory objects in the Identity Vault. To create a DS object:

1. Right-click on the Identity Vault, driver set, or driver in the modeler, then click **New** > **DS Object**.
2. Specify the name for the DS object.
3. Click the browse icon, then browse to and select the DS object you want to add to this Resource object.
4. (Conditional) If your authentication information is not saved in Designer, specify the following information to log into the Identity Vault:

   Host: Specify the IP address or DNS name of the server that contains your Identity Vault.

   Username: Specify the DN of a user object to authenticate to the Identity Vault. Specify the name using dot notation.

   Password: Specify the password of the user entered above.
5. After selecting the object, click **OK**.

6 Click Yes, to save the object so it can be opened in the editor.

7 Review the information in the XML editor, then close the editor.

Package Prompts

Package prompts are Resource objects that allow you to add prompts to packages you create. For more information, see "Adding Default Package Prompts" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Filters

Filters are Resource objects that allow you to add filters to packages. This object can only be created on driver objects and if the **Enable Package Developer Mode** option is selected on the Identity Vault.

To create a filter Resource object:

1 Right-click the Identity Vault, then click **Properties**.

2 Verify that the **Enable Package Developer Mode** option is selected, then close this page.

3 Right-click the driver, then click **New** > **Resource**.

4 Specify an name for the filter Resource object.

5 In the content type drop-down list, select **application/vnd.novell.dirxml.filter-ext+xml**, then click **OK**.

6 Click **Yes**, to save the object so it can be opened in the editor.

7 Add the desired information to the filter, then close the Filter Editor.

Library Objects

Packages replace and enhance the functionality that library objects provide. For more information about packages, see "Understanding Packages" in the *Designer Administration Guide for OpenText™ Identity Manager CE 24.4 (v4.10)* .

Library objects store multiple policies and other resources that are shared by one or more drivers. A library object can be created in a driver set object or any OpenText eDirectory container. Multiple libraries can exist in an OpenText eDirectory tree. Drivers can reference any library in the tree as long as the server running the driver holds a Read/Write or Master replica of the library object.

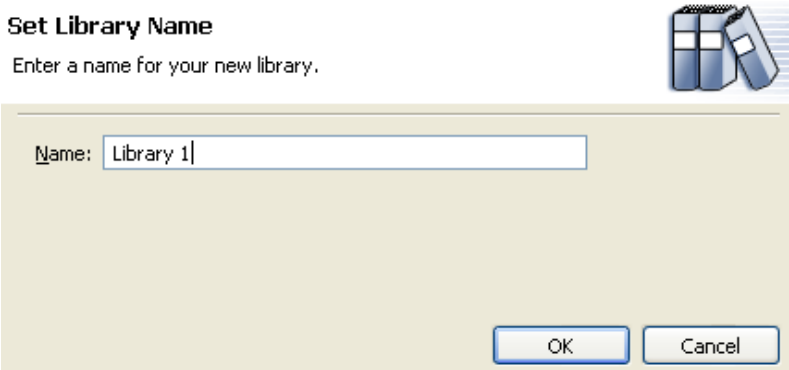
Style sheets, policies, rules, and other resource objects can be stored in a library and be referenced by one or more drivers.

- "Creating Library Objects" on page 171
- "Adding Policies to the Library Objects" on page 171
- "Using Policies in the Library Objects" on page 172

Creating Library Objects

1 Right-click a driver set or the Identity Vault object in the Outline view, then click **New > Library.**

2 Specify the name of the library object, then click **OK**.



Adding Policies to the Library Objects

Libraries can hold any policy, XSLT style sheets, or any type of resource object.

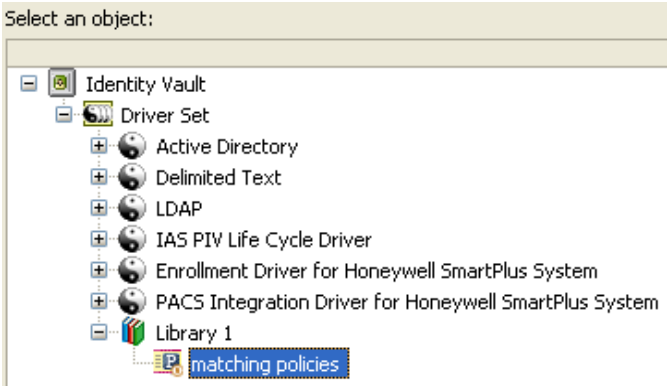
1 Right-click the library object, select **New**, then select the type of object you want stored in the library. The options are:

- **Credential Application:** Stores application authentication parameter values for Credential Provisioning policies. For information, see "Creating an Application Object" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.
- **Credential Repository:** Stores static configuration information for Credential Provisioning policies. For information, see "Creating a Repository Object" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.
- **DirXML Script:** Creates a policy set. See "Creating a Policy" on page 20 for more information.
- **ECMAScript:** Creates an ECMAScript object. See "Creating an ECMAScript Object" on page 173 for more information.
- **Mapping Table:** Creates a mapping table object. For more information, see "Creating a Mapping Table Object" on page 59.
- **Resource:** Creates a generic resource object. For more information, see "Creating a Generic Resource Object" on page 167.
- **Schema Map:** Creates a Schema Map object. For more information, see Chapter 5, "Defining Schema Map Policies," on page 75.
- **XSLT:** Creates an XSLT style sheet in the library. For more information, see "Defining Policies by Using XSLT Style Sheets" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.
- **From Copy:** Creates a copy of an existing object.

Using Policies in the Library Objects

After you have created the library, you can use any of the resources stored in the library in any policy.

1. Double-click the desired policy in the Outline view.
2. Right-click in the Policy Builder, then select **New > Include > Insert Include Before** or **Insert Include After**.
3. Browse to and select the desired resource stored in the library object, then click **OK** twice.



10 Using ECMAScript in Policies

ECMAScript is a scripting programming language, standardized by Ecma International. It is often referred to as JavaScript* or JScript, but these are actually implementations of ECMAScript. OpenText Identity Manager supports ECMAScript. ECMAScript objects are resource objects that store ECMAScripts. The ECMAScript is called through a policy to provide advanced functionality that DirXML Script or XSLT style sheets cannot provide.

Identity Manager uses the ECMACScript objects in two different ways: to create a custom form in the provisioning request definition editor, and to call an ECMAScript function in policies. For more information on custom forms, see Creating Forms for a Provisioning Request Definition in the *OpenText™ Identity Manager CE 24.4 (v4.10) - Administrator's Guide to Designing the Identity Applications*.

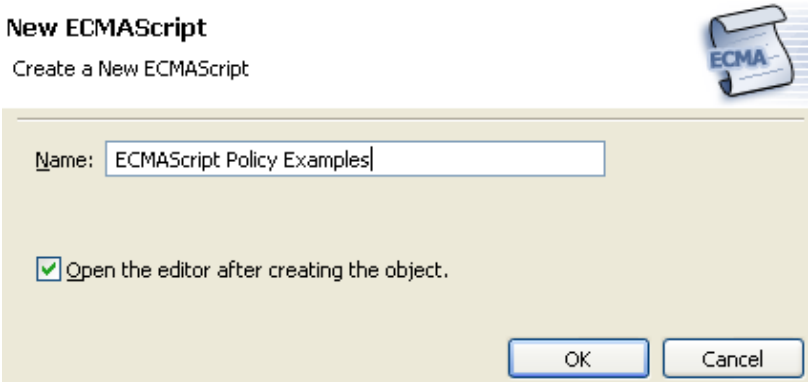
This section explains how to use the ECMAScript editor, how to use ECMAScript with policies, and how to use ECMAScript with custom forms. It does not explain the ECMAScript language. See ECMAScript Language Specification for information on how to use the ECMAScript language.

- "Creating an ECMAScript Object" on page 173
- "Using the ECMAScript Editor" on page 174
- "Examples of ECMAScripts with Policies" on page 183

Creating an ECMAScript Object

ECMAScript objects can be created in a library, driver object, Publisher channel, or Subscriber channel.

1. In the Outline view, right-click the location to create the ECMAScript object, then select **New > ECMAScript**.
2. Specify the name of the ECMAScript object.
3. Select the check box for **Open the editor after creating the object**, then click **OK**.



4. Click **Yes** in the file conflict message to save the ECMAScript object.

5 Either type the ECMAScript, or copy the ECMAScript into the editor from an existing file.

6 To save the ECMAScript press Ctrl+S after the ECMAScript is finished.

For information on how to use the ECMAScript editor, see Section 10, "Using ECMAScript in Policies," on page 173.

Using the ECMAScript Editor

ECMAScript objects are supported only with servers that have Identity Manager 3.5 or later installed. If a server in a selected driver set is earlier than Identity Manager 3.5, an error message is displayed, and Designer does not allow the object to be created. Change the version of the server to Identity Manager 3.5 or later on the properties of the server, then the ECMAScript object can be created.

Designer provides an ECMAScript editor, which also includes an ECMA Expression Builder. You use both to create the ECMAScript.

To access the ECMAScript editor:

1 Right-click an ECMAScript object in the Outline view, then select **Edit**.

   or

   When creating an ECMAScript object, select the check box **Open the editor after creating the object**.

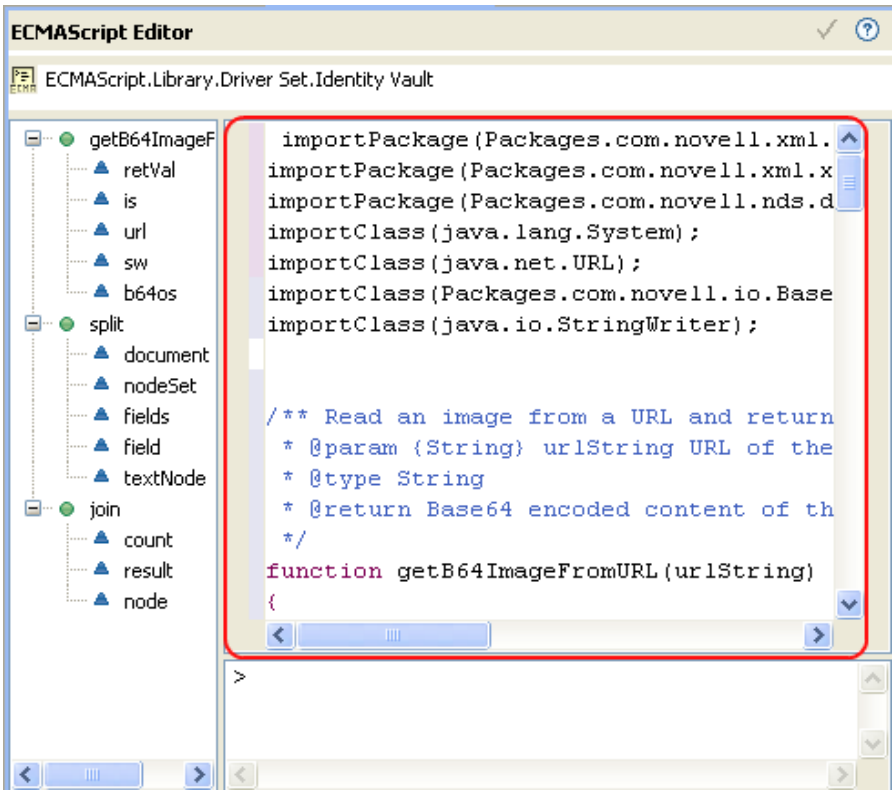
The ECMAScript editor provides different types of functionality depending upon which section you are using.

- "Main Scripting Area" on page 174
- "Expression Builder" on page 177
- "Functions and Variables" on page 179
- "Error Display" on page 180
- "Shell Area" on page 181

Main Scripting Area

The ECMAScript editor provides a main scripting area where the ECMAScript is created. You can type a new script, or copy an existing one.

Figure 10-1 *Main Scripting Area*



- "Using an Existing ECMAScript" on page 175
- "Editing an ECMAScript" on page 175
- "Coding Help for ECMAScript" on page 176

Using an Existing ECMAScript

1. Open the ECMAScript in a text editor, then copy the script.
2. Paste the ECMAScript into the ECMAScript editor.
3. Press Ctrl+S to save the ECMAScript.

Editing an ECMAScript

There are multiple options available for use to edit the ECMAScript.

Table 10-1 *ECMAScript Editing Options*

Option	Description
Undo Typing	Undoes the typing that has occurred.
Redo	Redoes the last action.
Cut	Cuts the selected area and adds it to the clipboard.

Option	Description
Copy	Copies the selected area into the clipboard.
Paste	Pastes the information in the clipboard into the main scripting area.
Delete	Deletes the selected information from the main scripting area.
Select All	Selectes all of the information in the main scripting area.
Find/Replace	Finds and replaces the specified information.
Show Expression Builder	Launches the Expression Builder. For more information, see "Expression Builder" on page 177.

Coding Help for ECMAScript

The ECMA Script editor contains coding helps. To access the coding helps, right-click in the left margin of the main scripting area, then select the desired option.

Table 10-2 *Coding Help*

Option	Description
Toggle Breakpoints	To be implemented.
Enable Breakpoints	Sets breakpoints in the ECMAScript.
Breakpoints Properties	Displays the properties of the breakpoints.
Add Bookmarks	Places a bookmark icon on a line in the ECMAScript editor.
Add Tasks	Places a task icon in a line as a reminder of additional work that needs to be done. If you open the Task view from the toolbar, by selecting **Window > Show View > Tasks**, the task is displayed.
Revert Block	To be implemented.
Delete Added Line	Deletes the last line added.
Show Quick Diff	To be implemented.
Show Line Numbers	Displays line numbers in the main scripting area.
Preferences	Sets the line delimitation and sets the suffix for the files created in the ECMAScript editor. By default, there is no translation for line delimiters, and the suffix is `js`.

Expression Builder

The Expression Builder helps in creating ECMAScript expressions. The Expression Builder can be accessed in two ways through the ECMAScript editor; it can also be accessed through the Policy Builder and the Argument Builder.

To access the Expression Builder in the ECMAScript editor:

1. Right-click in the main scripting area of the ECMAScript editor, then click **Show Expression Builder**.

   or

   Right-click the shell area of the ECMAScript editor, then click **Show Expression Builder**.

To access the Expression Builder through the Policy Builder:

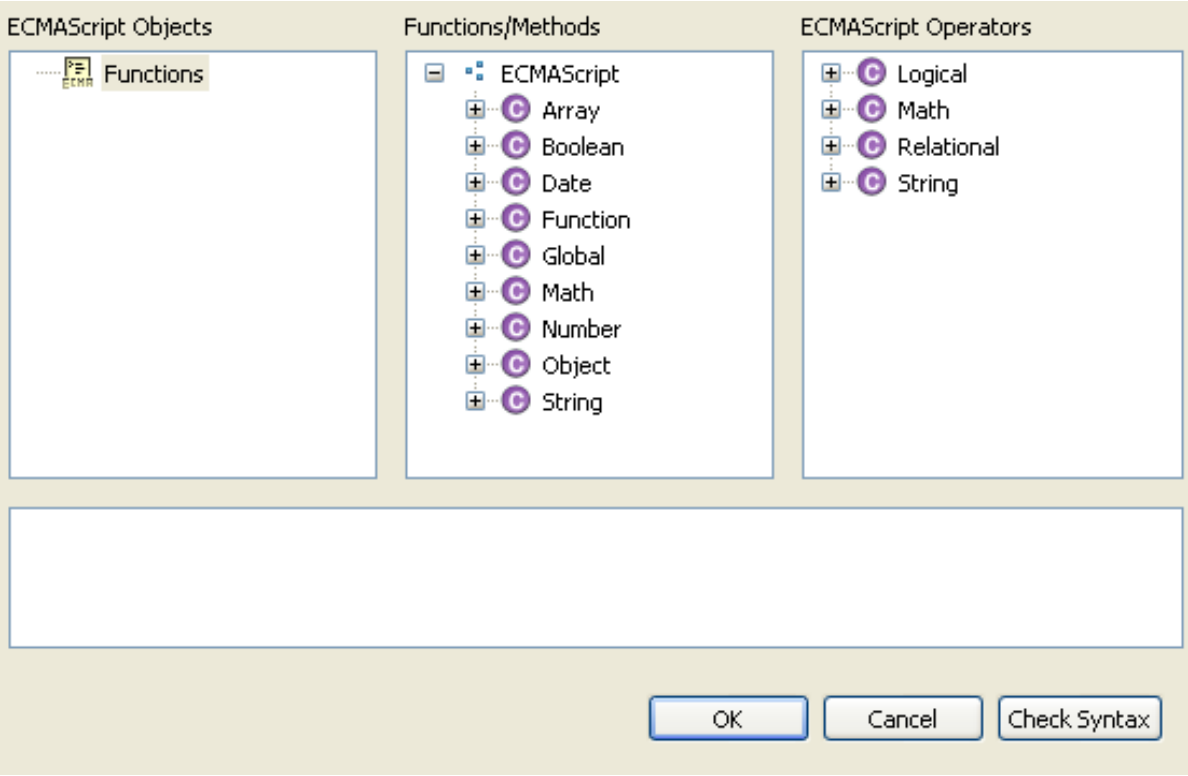
1. Click the **Launch ECMA Expression Builder** icon next to the following actions or conditions:.
   - If XPath Expression
   - Append XML Element
   - Append XML Text
   - Clone By XPath Expressions
   - Set XML Attribute
   - Strip XPath Expression

To access the Expression Builder through the Argument Builder:

1. Double click the XPath noun token.
2. Click the **Launch ECMA Expression Builder** icon in the Argument Builder.

The Expression Builder has three panes; **ECMAScript/Variables**, **Functions/Methods**, and **ECMAScript Operators**.

Figure 10-2 Expression Builder



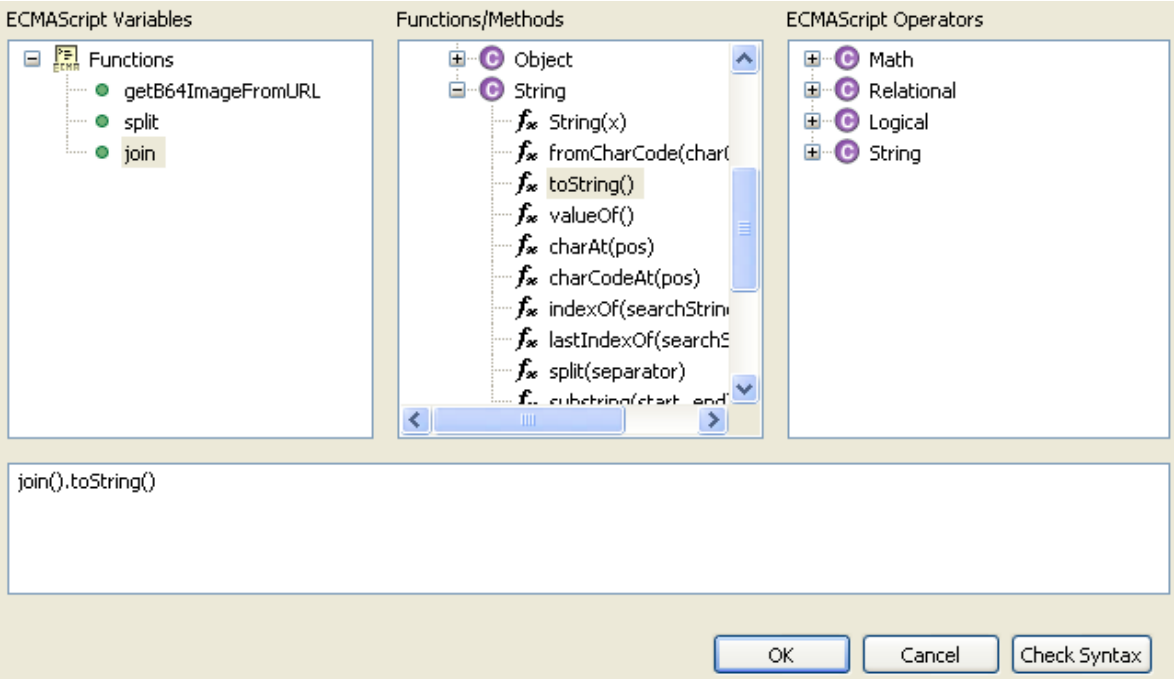
ECMAScript Objects lists all of the current defined functions in the ECMAScript. **Function/Methods** contains the standard ECMAScript functions. **ECMAScript Operators** displays the standard ECMAScript operators.

To use the Expression Builder:

1. (Optional) Click the desired **ECMAScript Objects.**
2. (Optional) Click the desired **Functions/Methods**.
3. (Optional) Click the desired **ECMAScript Operators.**
4. Click **Check Syntax** to validate the expression.
5. Click **OK** to close the Expression Builder.

In the following example, the join ECMAScript variable is used with the toString function or method, but there is no ECMAScript operator selected.

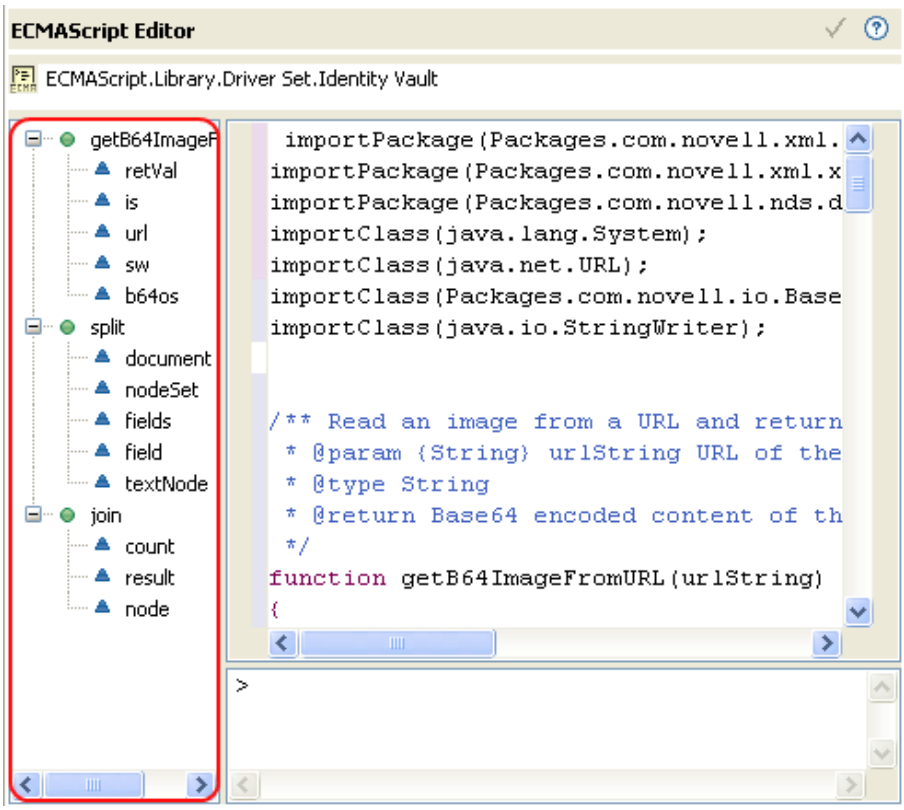
Figure 10-3 *Expression Builder Example*



Functions and Variables

As functions and variables are defined in the ECMAScript, they are displayed on the left side of the ECMAScript editor.

Figure 10-4 Functions and Variables



All of the variables that are stored in a function are grouped together. You can expand a function to view all of the variables, by clicking the plus icon (arrow icon in Linux). You can view the function without the variables by clicking the minus icon (arrow icon in Linux).

Error Display

As the ECMAScript is created, errors are displayed in the main scripting area and in the Problems view. The main scripting area displays the errors as a red X on the line where the error occurs.

Figure 10-5 Main Scripting Area Errors

```
    var count = 0;
    var result = "";
    // loop through the Nodes in the N
    for (var node = nodeSet.first(); n
    {
        // if not first Node, append t
        if (count++ > 0)
        {
            result += delimiter;
        }
        // append the string value of
        result += XPathUtil.getNodeVal
    }
    return result;
}
(a
```

The Problems view accumulates the errors as the ECMAScript is typed, displays the cause of the error.

Double-click the error in the Problems view. The cursor jumps to the problem line in the main scripting area.

To access the Problems view:

1 In the toolbar, select **Window > Show View > Other > General > Problems**.

  The Problems view is displayed below the ECMAScript editor.

Problems

1 error, 0 warnings, 0 infos

Description	Resource	Path	Location
Errors (1 item)			
missing) in parenthetical	B85O1672....	project/Model/IdentityMan...	line 122

Shell Area

The shell area of the ECMAScript area allows you execute the ECMAScript. After the ECMAScript is created, you can test the functionality of the script.

Figure 10-6 Shell Area

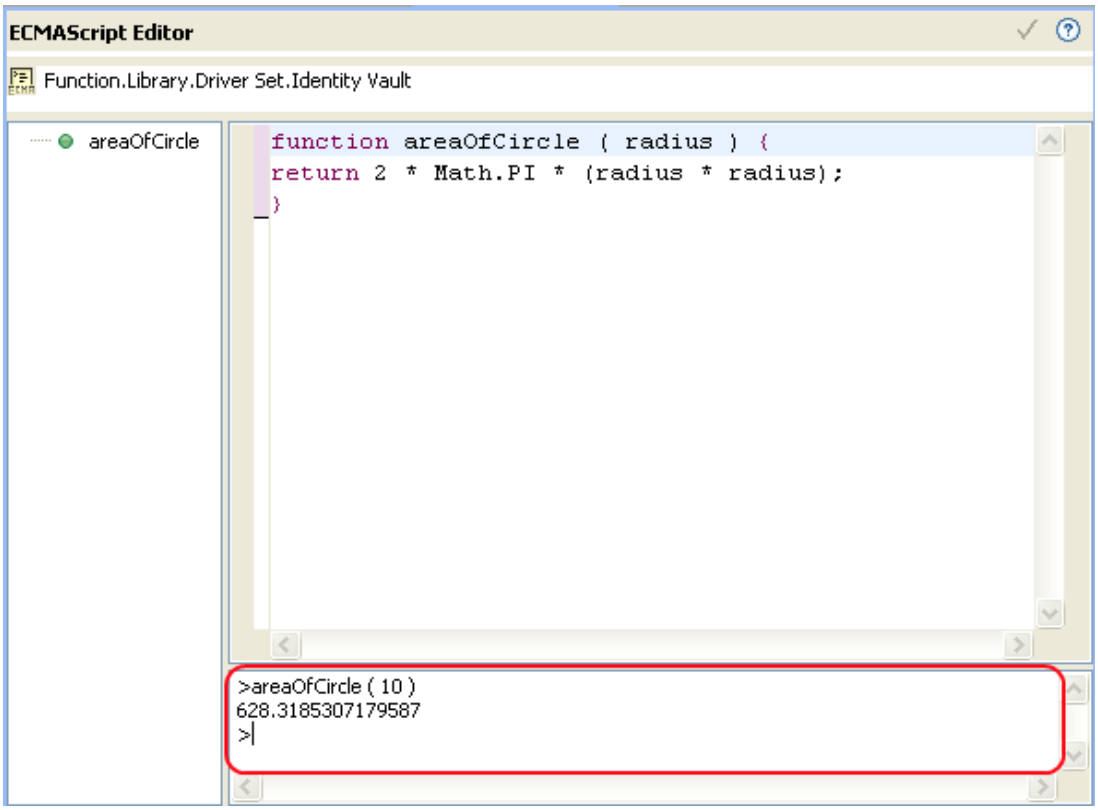


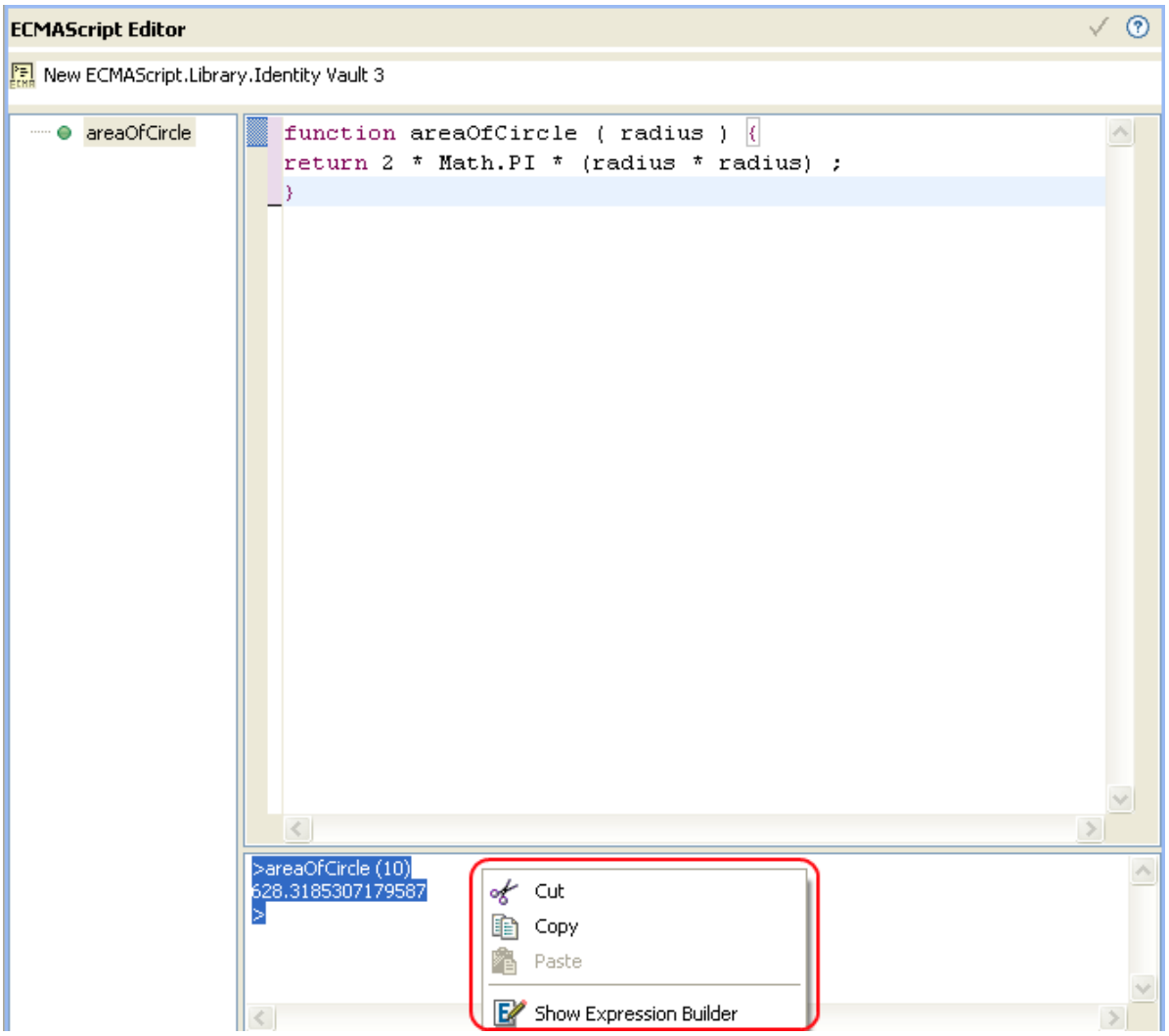
Figure 10-6 contains an example of a function that determines the area of a circle. The function is tested by specifying a value of `areaOfCircle(10)`. The shell displays the value of 628.3185307179587.

To execute the expression, press the Enter key. If you want to enter more than one line of code in the console, press Enter on the numeric keypad.

Additional Options in Shell Area

If you right-click inside the shell area you are presented with the following additional options:

Figure 10-7 Shell Area Additional Options



- **Cut, Copy and Paste:** Enables you to cut, copy, and paste from and into the shell area.
- **Show Expression Builder:** Launches ECMA Expression Builder.

Examples of ECMAScripts with Policies

The following examples use the ECMAScript file `demo.js` with different policies. The `demo.js` file contains three ECMAScript function definitions.

NOTE: To be able to call an ECMAScript function within a style sheet, ensure you include the following namespace definitions in the `xsl:stylesheet` element:

```
xmlns:js="http://www.novell.com/nxsl/ecmascript"
xmlns:es="http://www.novell.com/nxsl/ecmascript"
```


- "DirXML Script Policy Calling an ECMAScript Function" on page 184
- "XSLT Policy Calling an ECMAScript Function at the Driver Level" on page 185
- "XSLT Policy Calling an ECMAScript Function in the Style Sheet" on page 186

DirXML Script Policy Calling an ECMAScript Function

The DirXML Script policy converts an attribute that is a URL reference to a photo to the Base64 encoded photo data by calling the ECMAScript function getB64ImageFromURL (). The policy can be used as an Input Transformation or Output Transformation policy.

The function reads an image from a URL and returns the content as a Base64 encoded string.

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE policy PUBLIC "policy-
builder-dtd" "C:\Program
Files\Novell\Designer\eclipse\plugins\com.novell.designer.idm.policybuilde
r_1.2.0.200612180606\DTD\dirxmlscript.dtd"><policy>
  <rule>
    <description>Reformat photo from URL to octet</description>
    <conditions/>
    <actions>
      <do-reformat-op-attr name="photo">
        <arg-value type="octet">
          <token-xpath expression="es:getB64ImageFromURL(string($current-
value))"/>
        </arg-value>
      </do-reformat-op-attr>
    </actions>
  </rule>
</policy>
```

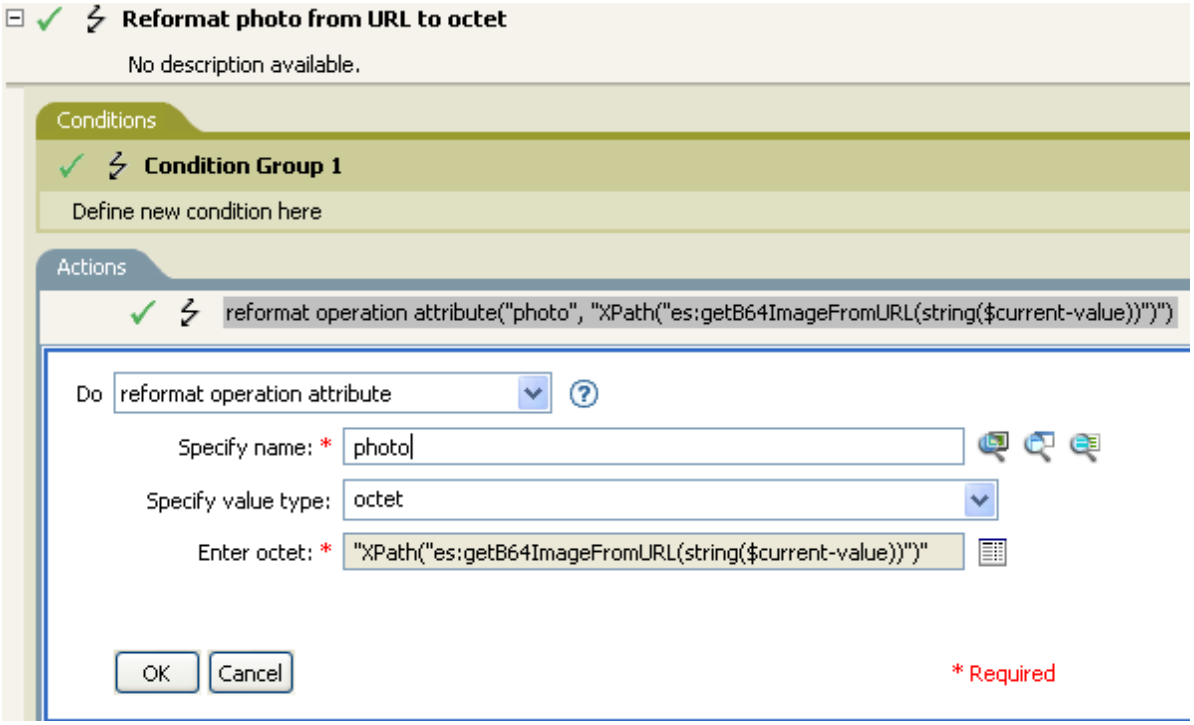
Function: `<static> String getB64ImageFromURL(<String> urlString)`

Parameters: urlString (URL of the image file)

Returns: Base64 encoded content of the image (or empty string if error)

The file `ReformatPhoto.xml` (../samples/ReformatPhoto.xml) calls the ECMAScript function getB64ImageFromURL from a DirXML Script policy. The file `phototest.xml` (../samples/phototest.xml) is a sample input document that shows the policy in action.

Figure 10-8 *Reformat Photo Example*



The ECMAScript calls the getB64ImageFromURL function, which then returns the current value as a string.

XSLT Policy Calling an ECMAScript Function at the Driver Level

The XSLT policy either splits a single comma-delimited value into multiple values, or joins multiple values into a single comma-delimited value. The XSLT policy is defined at the driver level and can be used as an Input Transformation or Output Transformation policy.

NOTE: DirXML Script has the split and join functionality built into it, but XSLT does not. This type of function allows XSLT to have the split and join functionality.

There are two functions:

- "Join" on page 185
- "Split" on page 186

Join

The Join function joins the text values of Nodes in a NodeSet into a single string

```
<!--  template that joins the joinme attribute values into a single value -
->
<xsl:template match="*[@attr-name='joinme']//*[value] | *[@attr-
name='joinme'][value]">
   <xsl:copy>
      <xsl:apply-templates select="@*|node()[not(self::value)]"/>
     <value>
        <xsl:value-of select="es:join(value)"/>
     </value>
   </xsl:copy>
</xsl:template>
```

Function: `<static> String join(<NodeSet> nodeSet, <string> delimiter)`

Parameters: nodeSet (the input NodeSet) and delimiter (the delimiter to split on. Optional: default = none).

Returns: The concatenation of the string values of the Nodes in the nodeSet, separated by the delimiter.

Split

The Split function splits a string into a NodeSet.

```
<!--  template that splits the splitme attribute values into multiple
values -->
<xsl:template match="*[@attr-name='splitme']//value">
     <xsl:for-each select="es:split(string(.))">
     <value>
        <xsl:value-of select="."/>
     </value>
     </xsl:for-each>
</xsl:template>
```

Function: `<static> NodeSet split(<String> inputString, <String> delimiter)`

Parameters: inputString (the script to split) and `delimiter` (the delimiter to split on. Optional: default = “,”).

Returns: A NodeSet containing text nodes.

The file `SplitJoin.xsl` (../samples/SplitJoin.xsl) calls the join or split functions in an XSLT style sheet. The file `splitjointest.xml` (../samples/splitjointest.xml) is an input document that shows the style sheet in action.

XSLT Policy Calling an ECMAScript Function in the Style Sheet

The XSLT policy demonstrates embedding ECMAScript function definitions with the XSLT style sheet. The functions convert a string to uppercase.

```
<!-- define ecmascript functions -->
<es:script>
function uppercase(input)
{
  return String(input).toUpperCase();
}
</es:script>
```

The file `uppercase.xsl` (../samples/uppercase.xsl) defines the ECMAScript function with the XSLT style sheet. The file `uppercasetest.xml` (../samples/uppercasetest.xml) is an input document that shows the style sheet in action.

11 Conditions

Conditions define when actions are performed. Conditions are always specified in either Conjunctive Normal Form (CNF) (http://mathworld.wolfram.com/ConjunctiveNormalForm.html) or Disjunctive Normal Form (DNF) (http://mathworld.wolfram.com/DisjunctiveNormalForm.html). These are logical expression forms. The actions of the enclosing rule are only performed when the logical expression represented in CNF or DNF evaluates to True or when no conditions are specified.

NOTE: For information about elements in the XML schema, see NDS DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

This section contains detailed information about all conditions that are available through the Policy Builder interface.

- "If Association" on page 190
- "If Attribute" on page 192
- "If Class Name" on page 195
- "If Destination Attribute" on page 198
- "If Destination DN" on page 201
- "If Entitlement" on page 203
- "If Global Configuration Value" on page 206
- "If Local Variable" on page 208
- "If Named Password" on page 212
- "If Operation" on page 213
- "If Operation Attribute" on page 216
- "If Operation Property" on page 220
- "If Password" on page 222
- "If Source Attribute" on page 225
- "If Source DN" on page 228
- "If XML Attribute" on page 230
- "If XPath Expression" on page 232

If Association

Performs a test on the association value of the current operation or the current object. The type of test performed depends on the operator specified by the operation attribute.

Fields

Operator

Operator	Returns True When...
Associated	There is an established association for the current object.
Available	There is a non-empty association value specified by the current operation.
Equal	The association value specified by the current operation is exactly equal to the content of the if association.
Greater Than	The association value specified by the current operation is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	The association value specified by the current operation is less than the content of the condition when compared by using the specified comparison mode.
Not Associated	There is not an established association for the current object.
Not available	The association is not available for the current object.
Not Equal	The association value specified by the current operation is not equal to the content of the if association.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

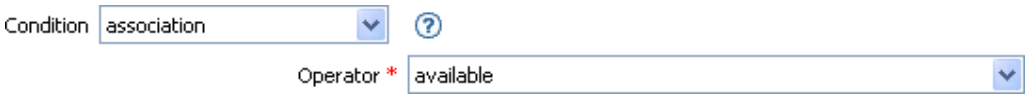
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example

This example tests to see if the association is available. When this condition is met, the actions that are defined are executed.



If Attribute

Performs a test on attribute values of the current object in either the current operation or the source data store. It can be logically thought of as If Operation Attribute or If Source Attribute, because the test is satisfied if the condition is met in the source data store or in the operation. The test performed depends on the specified operator.

Fields

Name

Specify the name of the attribute to test. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a value available in either the current operation or the source data store for the specified attribute.
Equal	There is a value available in either the current operation or the source data store for the specified attribute, which equals the specified value when compared by using the specified comparison mode.
Greater Than	There is a value available in either the current operation or the source data store for the specified attribute that is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is a value available in either the current operation or the source data store for the specified attribute that is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

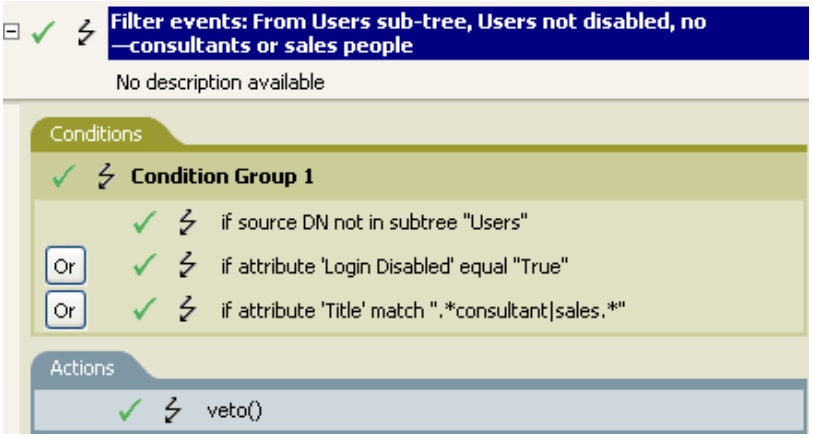
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.
Structured	Compares the structured attribute according to the comparison rules for the structured syntax of the attribute.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example

The example uses the condition If Attribute when filtering for User objects that are disabled or have a certain title. The policy is Policy to Filter Events, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Event-FilterByContainerDisabledOrTitle.xml` (../samples/001-Event-FilterByContainerDisabledOrTitle.xml).



The condition is looking for any User object that has an attribute of Title with a value of consultant or sales.

If Class Name

Performs a test on the object class name in the current operation.

Fields

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is an object class name available in the current operation.
Equal	There is an object class name available in the current operation, and it equals the specified value when compared by using the specified comparison mode.
Greater Than	There is an object class name available in the current operation, and it is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is an object class name available in the current operation, and it is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

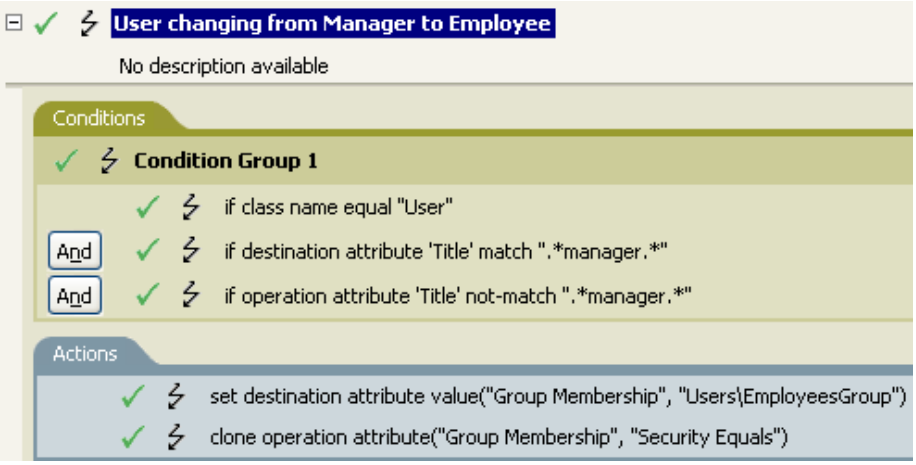
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison parameter are:

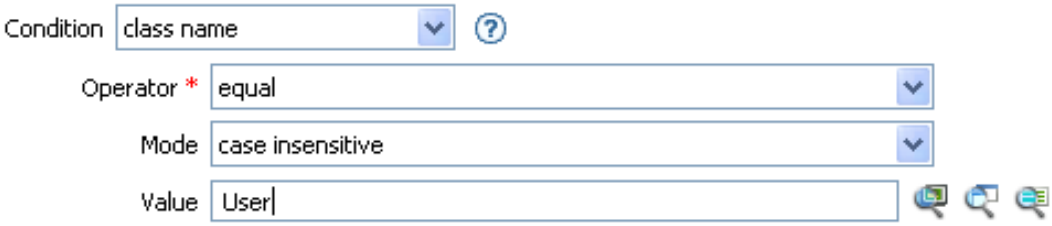
- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example

The example uses the condition If Class Name to govern group membership for a User object based on the title. The policy is Govern Groups for User Based on Title Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `004-Command-GroupChangeOnTitleChange.xml` (../samples/004-Command-GroupChangeOnTitleChange.xml).



Checks to see if the class name of the current object is User.



If Destination Attribute

Performs a test on attribute values of the current object or the object resolved using the specified `<arg-dn>` or `<arg-association>` in the destination data store. The test performed depends on the specified operator.

Fields

Name

Specify the name of the attribute to test. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a value available in the destination data store for the specified attribute.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared by using the specified comparison mode. If `mode="structured"` then the content must be a set of `<component>` elements, otherwise it must be text.
Greater Than	There is a value available for the specified attribute in the destination data store that is greater than the content of the condition when compared by using the specified comparison mode. If `mode="structured"`, the content must be a set of `<component>` elements; otherwise, it must be text.
Less Than	There is a value available for the specified attribute in the destination data store that is greater than the content of the condition when compared by using the specified comparison mode. If `mode="structured"`, the content must be a set of `<component>` elements; otherwise, it must be text.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Great Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than

- Not Equal
- Not Greater Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.
Structured	Compares the structured attribute according to the comparison rules for the structured syntax of the attribute.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Token

Allows you to select a DN or an association.

Operation Token

Allows you to select the proper value of the DN or the association with the Object selector or variable browser.

Example

The policy checks to see if the value of the title attribute contains manager.

Condition destination attribute

Class User

Name * Surname

Operator * equal

Mode case insensitive

Value x2

Token dn

Operation Token cn=x2,ou=users,o=data

OK Cancel

If Destination DN

Performs a test on the destination DN in the current operation. The test performed depends on the specified operator.

Fields

Operator

Select the condition test type.

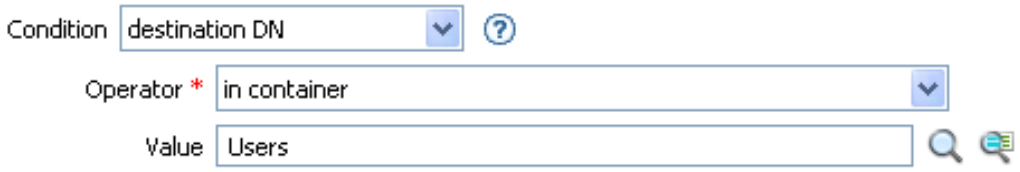
Operator	Returns True When...
Available	There is a destination DN available.
Equal	There is a destination DN available, and it equals the specified value when compared by using semantics appropriate to the DN format of the destination data store.
in Container	There is a destination DN available, and it represents an object in the container, specified by value, when compared by using semantics appropriate to the DN format of the destination data store.
In Subtree	There is a destination DN available, and it represents an object in the subtree, specified by value, when compared by using semantics appropriate to the DN format of the destination data store.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not in Container	In Container would return False.
Not In Subtree	In Subtree would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- In Container
- In Subtree
- Not Equal
- Not in Container
- Not in Subtree

Example


Condition
destination DN
Operator *
in container
Value
Users

If Entitlement

Performs a test on entitlements of the current object, in either the current operation or the Identity Vault. The test performed depends on the specified operator.

Fields

Name

Specify the name of the entitlement to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	The named entitlement is available in either the current operation or the Identity Vault.
Changing	The current operation contains a change (modify attribute or add attribute) of the named entitlement.
Changing From	The current operation contains a change that removes a value (remove value) of the named entitlement, which has a value that equals the specified value when compared by using the specified comparison mode.
Changing To	The current operation contains a change that adds a value (add value or add attribute) to the named entitlement. It has a value that equals the specified value when compared by using the specified comparison mode.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared by using the specified comparison mode.
Greater Than	The named entitlement is available and granted in either the current operation or the Identity Vault and has a value that is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	The named entitlement is available and granted in either the current operation or the Identity Vault and has a value that is less than the content of the condition when compared by using the specified comparison mode.
Not available	Available would return False.
Not Changing	Changing would return False.
Not Changing From	Changing From would return False.
Not Changing To	Changing To would return False.
Not Equal	Equal would return False.

Operator	Returns True When...
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Changing From
- Changing To
- Equal
- Greater Than
- Less Than
- Not Changing From
- Not Changing To
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Changing From
- Changing To

- Equal
- Greater Than
- Less Than
- Not Changing From
- Not Changing To
- Not Equal
- Not Greater Than
- Not Less Than

Example

If Global Configuration Value

Performs a test on a global configuration value. The test performed depends on the specified operator.

Remark

For more information on using variables with policies, see "Understanding Policy Components" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Fields

Name

Specify the name of the global value to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a global configuration value with the specified name.
Equal	There is a global configuration value with the specified name, and its value equals the specified value when compared by using the specified comparison mode.
Greater Than	There is a global configuration value with the specified name, and its value is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is a global configuration value with the specified name, and its value is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal

- Not Greater Than
- Not Less Than

Mode

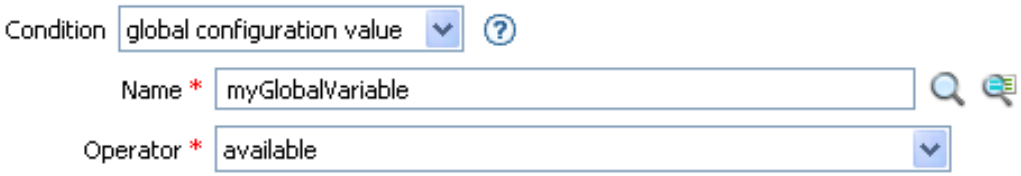
The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example



If Local Variable

Performs a test on a local variable. The test performed depends on the specified operator.

Remark

For more information on using variables with policies, see "Understanding Policy Components" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Fields

Name

Specify the name of the local variable to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a local variable with the specified name that has been defined by an action of a earlier rule within the policy.
Equal	There is a local variable with the specified name, and its value equals the specified value when compared by using the specified comparison mode.
Greater Than	There is a local variable with the specified name, and its value is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is a local variable with the specified name, and its value is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less than or equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal

- Not Greater Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

The example adds a User object to the appropriate Employee or Manager group based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy is Govern Groups for User Based on Title Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-AddCreate-Groups.xml` (../samples/003-Command-AddCreateGroups.xml).

Set local variables to test existence of groups and for placement

Create ManagersGroup, if needed

No description available

Conditions

Condition Group 1

if local variable 'manager-group-info' available

And if local variable 'manager-group-info' not equal "group"

Actions

add destination object(class name="Group", when="before", dn(Local Variable("manager-group-dn")))

Create EmployeesGroup, if needed

If Title indicates Manager, add to ManagerGroup and set rights

If Title does not indicate Manager, add to EmployeeGroup and set rights

The policy contains five rules that are dependent on each other.

Set local variables to test existence of groups and for placement

No description available

Conditions

Condition Group 1

if class name equal "User"

And

Condition Group 2

if operation equal "add"

Or if operation equal "modify"

Actions

set local variable("manager-group-dn", "Users\ManagersGroup")

set local variable("manager-group-info", Destination Attribute("Object Class", dn(Local Variable("manager-group-dn"))))

set local variable("employee-group-dn", "Users\EmployeesGroup")

set local variable("employee-group-info", Destination Attribute("Object Class", dn(Local Variable("employee-group-dn"))))

For the If Locate Variable condition to work, the first rule sets four different local variables to test for groups and where to place the groups.

Condition local variable

Name * manager-group-info

Operator * not equal

Mode case insensitive

Value group

The condition the rule looks for is to see if the local variable of manager-group-info is available and if manager-group-info is not equal to group. If these conditions are met, then the destination object of group is added.

If Named Password

Performs a test on a named password from the driver in the current operation with the specified name. The test performed depends on the selected operator.

Fields

Name

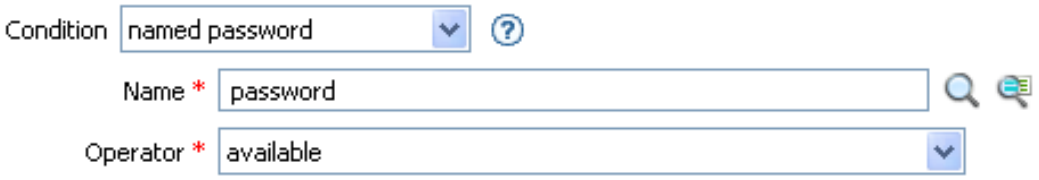
Specify the name of the named password to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a password with the specified name available.
Not Available	Available would return False.

Example



If Operation

Performs a test on the name of the current operation. The type of test performed depends on the specified operator.

Fields

Operator

Select the condition test type.

Operator	Returns True When...
Equal	The name of the current operation is equal to the content of the condition when compared by using the specified comparison mode.
Greater Than	The name of the current operation is greater than content of the condition when compared by using the specified comparison mode.
Less Than	The name of the current operation is less than content of the condition when compared by using the specified comparison mode.
Not Equal	Equal would return False.
Not Greater Than	Greater Than would return False.
Not Less Than	Less Than would return False.

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal

- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

The values are the operations that the OpenText Identity Manager engine looks for:

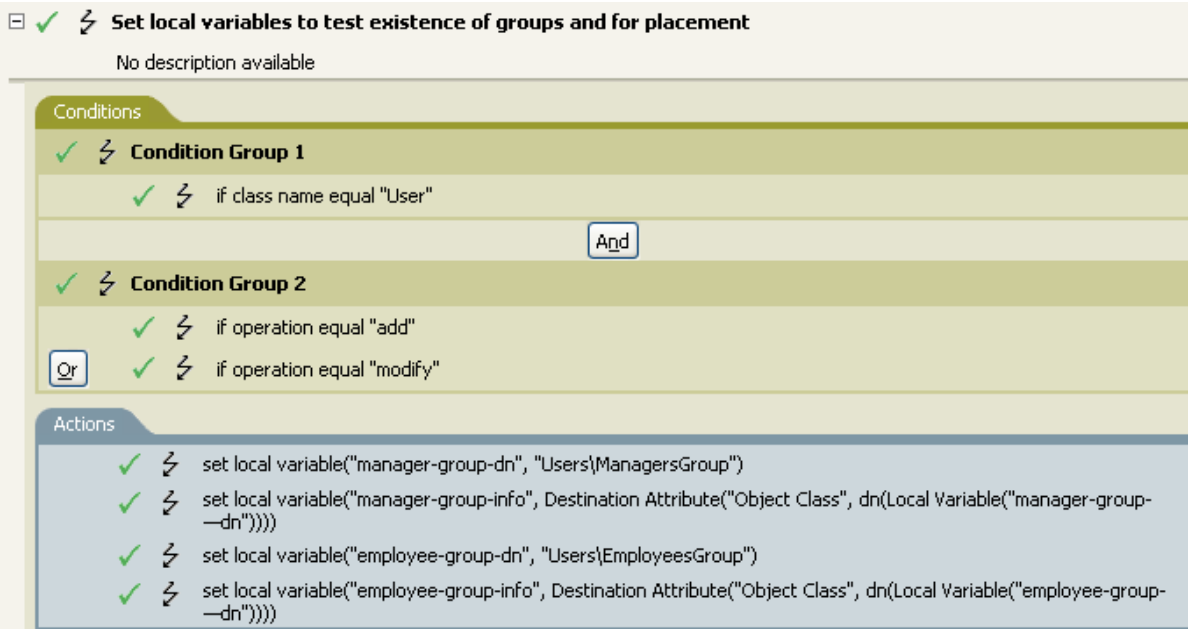
- add
- add-association
- check-object-password
- check-password
- delete
- generated-password
- get-named-password
- init-params
- instance
- modify
- modify-association
- modify-password
- move
- password
- query
- query-schema
- remove-association
- rename
- schema-def
- status

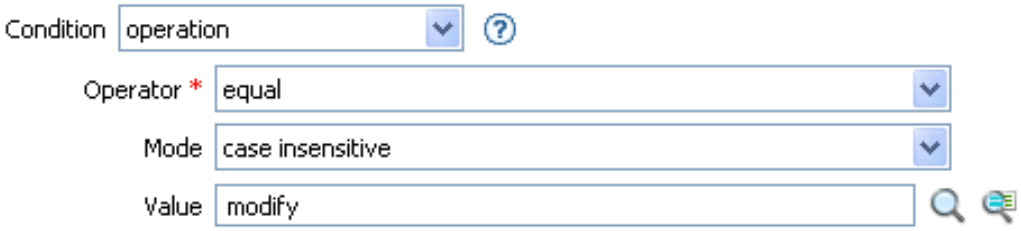
- sync
- trigger

This list is not exclusive. Custom operations can be implemented by drivers and administrators.

Example

The example adds a User object to the appropriate Employee or Manager group based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy name is Govern Groups for User Based on Title Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).





The condition checks to see if an Add or Modify operation has occurred. When one of these occurs, it sets the local variables.

If Operation Attribute

Performs a test on attribute values in the current operation. The test performed depends on the specified operator.

Fields

Name

Specify the name of the attribute to test. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a value available in the current operation (`<add-attr>`, `<add-value>` or `<attr>`) for the specified attribute. Available would return False if there is only a `<remove-value>` or a `<remove-all-values>` value for the specified attribute.
Changing	The current operation contains a change (`<modify-attr>` or `<add-attr>`) of the specified attribute. Changing would return True if it is an `<add-attr>`, <add-value>, `<remove-value>`, or `<remove-all-values>` operation for the specified attribute.
Changing From	The current operation contains a change that removes a value (`<remove-value>`) of the specified attribute that equals the content of the condition when compared by using the specified comparison mode. If `mode=structured`, then the content must be a set of `<component>'s`. Otherwise, it must be text.
Changing To	The current operation contains a change that adds a value (`<add-value>` or `<add-attr>`) to the specified attribute that equals the content of the condition when compared by using the specified comparison mode. If `mode=structured,` then the content must be a set of `<component>'s.` Otherwise, it must be text.
Equal	There is a value available in the current operation (other than a `<remove-value>`) for the specified attribute that equals the content of the condition when compared by using the specified comparison mode. If `mode=structured`, then the content must be a set of `<component>'s`. Otherwise, it must be text. Supports variable expansion.

Operator	Returns True When...
Greater Than	There is a value available in the current operation (other than a `<remove-value>`) for the specified attribute that is greater than the content of the condition when compared by using the specified comparison mode. If `mode=structured`, then the content must be a set of `<component>'s`. Otherwise, it must be text. Supports variable expansion.
Less Than	There is a value available in the current operation (other than a `<remove-value>`) for the specified attribute that is less than the content of the condition when compared by using the specified comparison mode. If `mode=structured`, then the content must be a set of `<component>'s`. Otherwise, it must be text. Supports variable expansion.
Not Available	Available would return False.
Not Changing	Changing would return False.
Not Changing From	Changing From would return False.
Not Changing To	Changing To would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Changing From
- Changing To
- Equal
- Greater Than
- Less Than
- Not Changing From
- Not Changing To
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

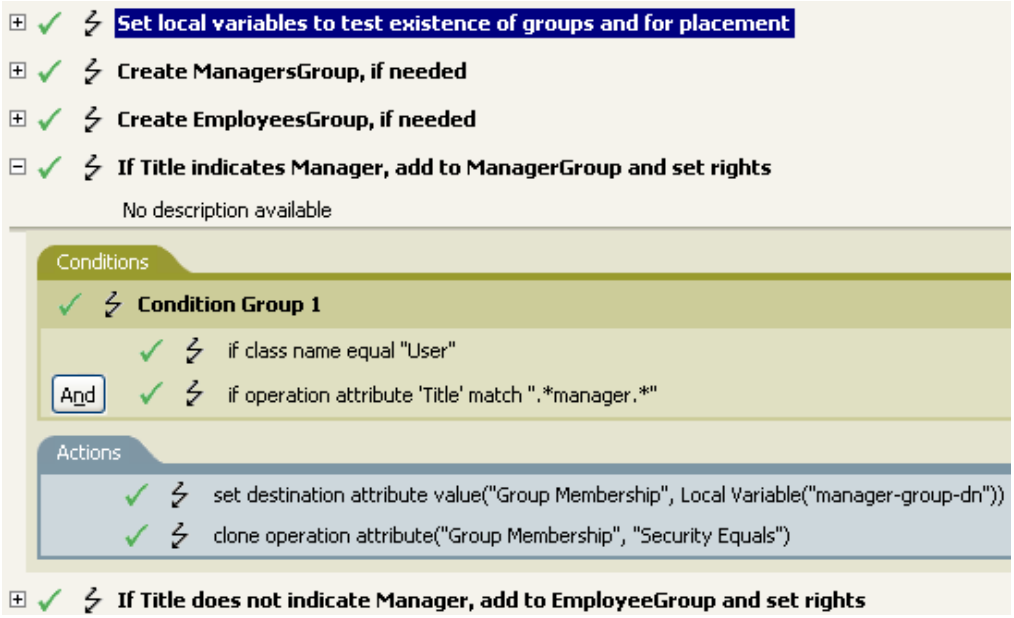
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.
Structured	Compares the structured attribute according to the comparison rules for the structured syntax of the attribute.

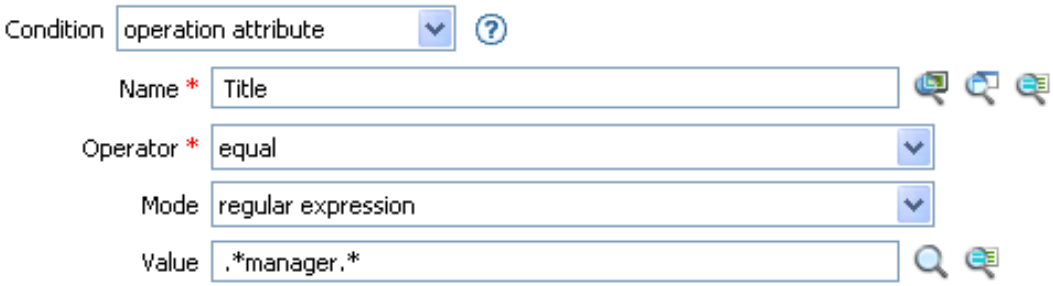
The operators that contain the comparison mode parameter are:

- Changing From
- Changing To
- Equal
- Greater Than
- Less Than
- Not Changing From
- Not Changing To
- Not Equal
- Not Greater Than
- Not Less Than

Example

The example adds a User object to the appropriate Employee or Manager group based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy name is Govern Groups for User Based on Title Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-Add-CreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).





The condition checks to see if the attribute of Title is equal to `.*manager.*`, which is a regular expression. The condition looks for a title that has zero or more characters before manager and a single character after manager. It would find a match if the User object's title was sales managers.

If Operation Property

Performs a test on an operation property on the current operation. An operation property is a named value that is stored as an attribute on an `<operation-data>` element within an operation. It is typically used to supply additional context that might be needed by the policy that handles the results of an operation. The test performed depends on the selected operator.

Fields

Name

Specify the name of the operation property to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is an operation property with the specified name on the current operation.
Equal	There is an operation property with the specified name on the current operation, and its value equals the provided content when compared by using the specified comparison mode.
Greater Than	There is an operation property with the specified name on the current operation, and its value is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is an operation property with the specified name on the current operation, and its value is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

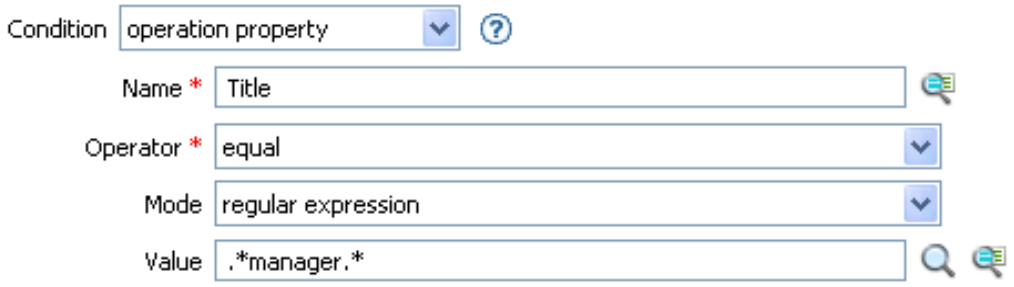
The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example



If Password

Performs a test on a password in the current operation. The test performed depends on the specified operator.

Fields

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a password available in the current operation.
Equal	There is a password available in the current operation, and its value equals the content of the condition when compared by using the specified comparison mode.
Greater Than	There is a password available in the current operation, and its value is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	There is a password available in the current operation, and its value is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example

If you are implementing Credential Provisioning policies, there is a sample Subscriber Command Transformation policy that uses the password condition. The sample file is called `SampleSubCommandTransform.xml`. It is found in the `\dirxml\utilities` folder on the OpenText Identity Manager media. For more information, see "Example Credential Provisioning Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. To view the policy in XML, see `SampleSubCommandTransform.xml` (../samples/SampleSubCommandTransform.xml).

The Subscriber Command Transformation policy checks to see if a password is available when an object is added. If the password is available, then the SecureLogin and SecretStore credentials are provisioned.

Add operation-data element to password subscribe operations (if needed)

Add payload data to modify-password subscribe operations

Add payload data to add subscribe operations

No description available

Conditions

Condition Group 1

if operation equal "add"

And if password available

Actions

```
append XML element("sso-sync-data", "operation-data")
append XML element("sso-target-user-dn", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-target-user-dn", Source Attribute("DirXML-ADContext"))
append XML element("sso-app-username", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-app-username", Source Attribute("CN"))
append XML element("password", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/password", Password())
append XML element("nsl-set-passphrase-answer", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/nsl-set-passphrase-answer", Source Attribute("workforceID"))
```

Condition password

Operator * available

If Source Attribute

Performs a test on attribute values of the current object or the object resolved using the specified <arg-dn> or <arg-association> in the source data store. The test performed depends on the specified operator.

Fields

Name

Specify the name of the source attribute to test for the selected condition. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a value available in the source data store for the specified attribute.
Equal	There is a value available in the source data store for the specified attribute. It equals the specified value when compared by using the specified comparison mode. If mode is structured then the content must be a set of components; otherwise, it must be text.
Greater Than	There is a value available in the source data store for the specified attribute that is greater than the content of the condition when compared by using the specified comparison mode. If the mode is structured, the content must be a set of components; otherwise, it must be text.
Less Than	There is a value available in the source data store for the specified attribute that is less than the content of the condition when compared by using the specified comparison mode. If the mode is structured, the content must be a set of components; otherwise, it must be text.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Great Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal

- Not Greater Than
- Not Less Than

Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.
Structured	Compares the structured attribute according to the comparison rules for the structured syntax of the attribute.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

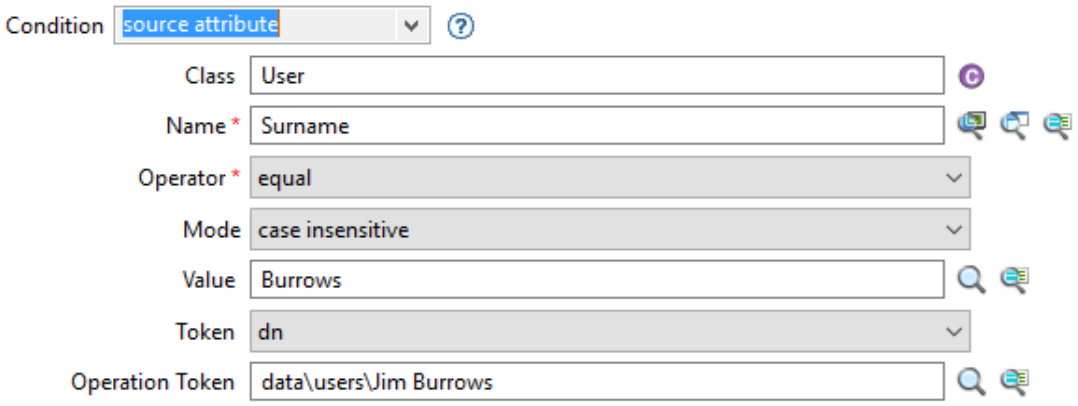
Token

Allows you to select a DN or an association.

Operation Token

Allows you to select the proper value of the DN or the association with the Object selector or variable browser.

Example


Condition
source attribute
Class
User
Name *
Surname
Operator *
equal
Mode
case insensitive
Value
Burrows
Token
dn
Operation Token
data\users\Jim Burrows

If Source DN

Performs a test on the source DN in the current operation. The test performed depends on the specified operator.

Fields

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is a source DN available.
Equal	There is a source DN available, and it equals the content of the specified value in-container.
In Container	There is a source DN available, and it represents an object in the container specified by the content of If Source DN, when compared by using semantics appropriate to the DN format of the source data store.
In Subtree	There is a source DN available, and it represents an object in the subtree identified by the specified value.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not In Container	In Container would return False.
Not In subtree	In Subtree would return False.

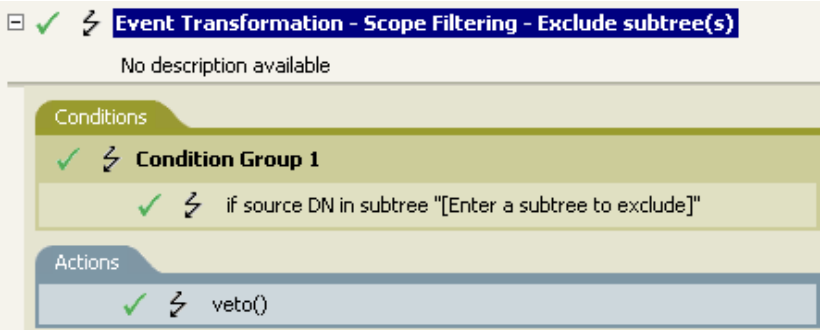
Value

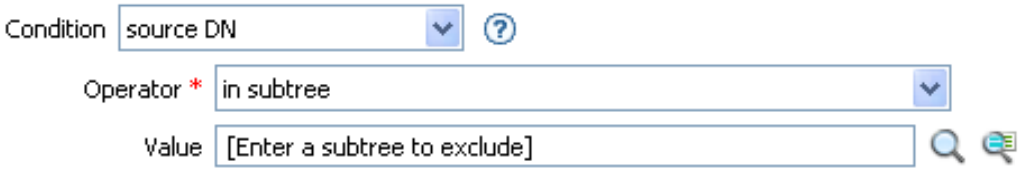
Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- In Container
- In Subtree
- Not Equal
- Not in Container
- Not in Subtree

Example

The example uses the condition If Source DN to check if the User object is in the source DN. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Event Transformation - Scope Filtering - Exclude Subtrees" on page 120. To view the policy in XML, see `predef_transformation_filter_exclude_subtress.xml` (../samples/predef_transformation_filter_exclude_subtrees.xml).





The condition checks to see if the source DN is in the Users container. If the object comes from that container, it is vetoed.

If XML Attribute

Performs a test on an XML attribute of the current operation. The type of test performed depends on the operator specified by the operation attribute.

Fields

Name

Specify the name of the XML attribute. An XML attribute is a name/value pair associated with an element in an XDS document.

Operator

Select the condition test type.

Operator	Returns True When...
Available	There is an XML attribute with the specified name on the current operation.
Equal	There is a an XML attribute with the specified name on the current operation, and its value equals the content of the condition when compared by using the specified comparison mode.
Greater Than	There is a an XML attribute with the specified name on the current operation, and its value is greater than the content of the condition when compared by using the specified comparison mode.
Less Than	The association value specified by the current operation is less than the content of the condition when compared by using the specified comparison mode.
Not Available	Available would return False.
Not Equal	Equal would return False.
Not Greater Than	Greater Than or Equal would return False.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information, see "Variable Selector" on page 31. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Mode

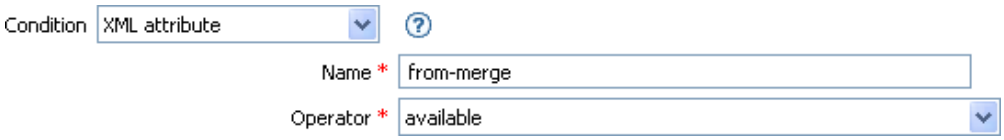
The condition has a comparison mode parameter that indicates how a comparison is done.

Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. For more information, see the Oracle Java documentation. The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.
Source DN	Compares by using semantics appropriate to the DN format for the source data store.
Destination DN	Compares by using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

The operators that contain the comparison mode parameter are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

Example



If XPath Expression

Performs a test on the results of evaluating an XPath 1.0 expression.

Fields

Operator

Select the condition test type.

Operator	Returns True When...
True	The XPath expression evaluates to True.
Not True	True would return False.

Value

Contains the value defined for the selected operator. The value is used by the condition. Each value supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35. The operators that contain the value field are:

- Equal
- Greater Than
- Less Than
- Not Equal
- Not Greater Than
- Not Less Than

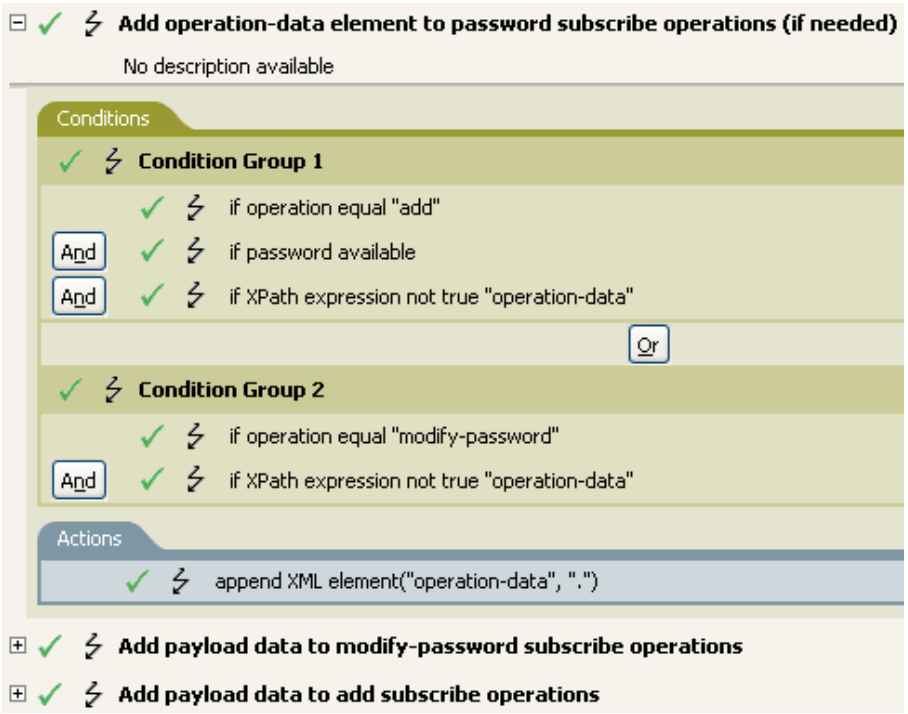
Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example

If you are implementing Credential Provisioning policies, there is a sample Subscriber Command Transformation policy that uses the XPath Expression condition. The sample file is called `SampleSubCommandTransform.xml`. It is found in the `\dirxml\utilities` folder on the OpenText Identity Manager media. For more information, see "Example Credential Provisioning Policies" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. To view the policy in XML, see `SampleSubCommandTransform.xml` (../samples/SampleSubCommandTransform.xml).

The sample Credential Provisioning policy checks each Add operation to see if there is operation data associated with the Add. If there is no operation data, the SecureLogin and SecretStore credentials are provisioned.


Add operation-data element to password subscribe operations (if needed)
No description available
Conditions
Condition Group 1
if operation equal "add"
And
if password available
And
if XPath expression not true "operation-data"
Or
Condition Group 2
if operation equal "modify-password"
And
if XPath expression not true "operation-data"
Actions
append XML element("operation-data", ".")
Add payload data to modify-password subscribe operations
Add payload data to add subscribe operations

Condition
XPath expression
Operator *
not true
Value
operation-data

12 Actions

Policies perform actions when the associated conditions are met. Some actions have a **Mode** field. The policy does not honor the mode at run time if the context in which the policy is running is incompatible with the selected mode.

NOTE: For information about elements in the XML schema, see NDS DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

This section contains detailed information about the actions available in the Policy Builder interface:

- "Add Association" on page 238
- "Add Destination Attribute Value" on page 239
- "Add Destination Object" on page 241
- "Add Resource" on page 243
- "Add Role" on page 245
- "Add Source Attribute Value" on page 247
- "Add Source Object" on page 248
- "Append XML Element" on page 249
- "Append XML Text" on page 251
- "Break" on page 253
- "Clear Destination Attribute Value" on page 254
- "Clear Operation Property" on page 255
- "Clear Source Attribute Value" on page 256
- "Clear SSO Credential" on page 257
- "Clone By XPath Expressions" on page 258
- "Clone Operation Attribute" on page 259
- "Create Resource" on page 261
- "Create Role" on page 264
- "Delete Destination Object" on page 266
- "Delete Source Object" on page 267
- "Delete Resource" on page 268
- "Delete Role" on page 269
- "Find Matching Object" on page 270
- "For Each" on page 273
- "Generate Event" on page 274
- "If" on page 277

- “If Operation Entitlement” on page 279
- “Implement Entitlement” on page 281
- “Invoke REST Endpoint” on page 282
- “Modify Resource” on page 284
- “Modify Role” on page 286
- “Move Destination Object” on page 288
- “Move Source Object” on page 290
- “Reformat Operation Attribute” on page 291
- “Remove Association” on page 293
- “Remove Destination Attribute Value” on page 294
- “Remove Named Password” on page 295
- “Remove Role” on page 296
- “Remove Resource” on page 298
- “Remove Source Attribute Value” on page 300
- “Rename Destination Object” on page 301
- “Rename Operation Attribute” on page 302
- “Rename Source Object” on page 303
- “Send Email” on page 304
- “Send Email from Template” on page 307
- “Set Default Attribute Value” on page 310
- “Set Destination Attribute Value” on page 312
- “Set Destination Password” on page 314
- “Set Local Variable” on page 315
- “Set Named Password” on page 317
- “Set Operation Association” on page 318
- “Set Operation Class Name” on page 319
- “Set Operation Destination DN” on page 320
- “Set Operation Property” on page 321
- “Set Operation Source DN” on page 322
- “Set Operation Template DN” on page 323
- “Set Source Attribute Value” on page 324
- “Set Source Password” on page 326
- “Set SSO Credential” on page 327
- “Set SSO Passphrase” on page 328
- “Set XML Attribute” on page 329
- “Start Workflow” on page 330
- “Status” on page 332
- “Strip Operation Attribute” on page 333

- “Strip XPath Expression” on page 334
- “Trace Message” on page 335
- “Veto” on page 337
- “Veto If Operation Attribute Not Available” on page 338
- “While” on page 339

This section contains detailed information about all actions available in the Policy Builder.

Add Association

Sends an `add association` command with the specified association to the Identity Vault.

Fields

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

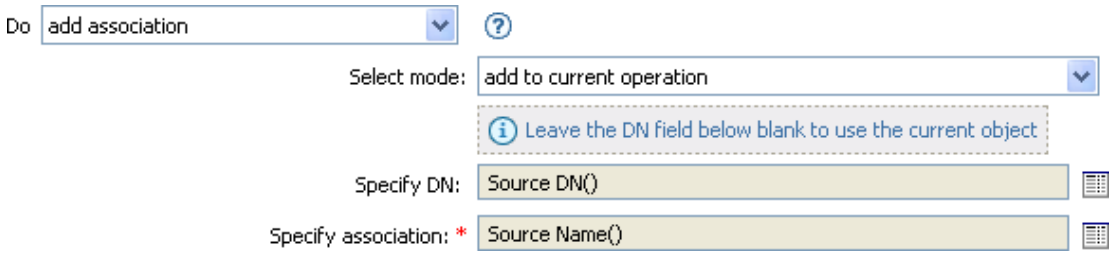
DN

Specify the DN of the target object or leave the field blank to use the current object.

Association

Specify the value of the association to be added.

Example



Add Destination Attribute Value

Adds a value to an attribute on an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN

Specify the DN, association, or current object as the target object.

Value Type

Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

Value

Specify the attribute value to be added.

Example

The example adds the destination attribute value to the OU attribute. It creates the value from the local variables that are created. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Create Departmental Container - Part 1 and Part 2" on page 108. To see the policy in XML, see `predef_command_create_dept_container1.xml` (../samples/predef_command_create_dept_container1.xml) and `predef_command_create_dept_container2.xml` (../samples/predef_command_create_dept_container2.xml).

Command Transformation - Create Departmental Container - Part 1

No description available

Conditions

Condition Group 1

if operation equal "add"

Actions

set local variable("target-container", Destination DN(length="-2"))

set local variable("does-target-exist", Destination Attribute —("objectclass", class name="Organizational Unit", dn(Local —Variable("target-container"))))

Command Transformation - Create Departmental Container - Part 2

No description available

Conditions

Condition Group 1

if local variable 'does-target-exist' available

And

if local variable 'does-target-exist' equal ""

Actions

add destination object(class name="Organizational Unit", direct="—true", dn(Local Variable("target-container")))

add destination attribute value("ou", direct="true", dn(Local —Variable("target-container")), Parse DN("dest-dn", "dot", length="—1", start="-1", Local Variable("target-container")))

Do: add destination attribute value

Specify attribute name: * CN

Specify class name:

Select mode: write directly to destination datastore

Select object: DN

Specify DN: * "Local Variable("target-containter")"

Specify value type: string

Enter string: * "Parse DN("dest-dn", "dot", length="1", start="1", Local Variat

Add Destination Object

Creates an object of the specified type in the destination data store, with the name and location specified in the **Enter DN** field. Any attribute values to be added as part of the object creation must be done in subsequent Add Destination Attribute Value actions, using the same DN.

Fields

Class Name

Specify the class name of the object to be created. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

DN

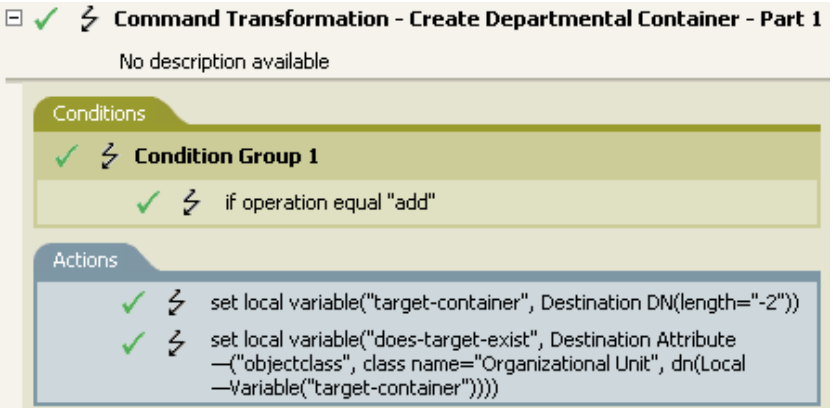
Specify the DN of the object to be created.

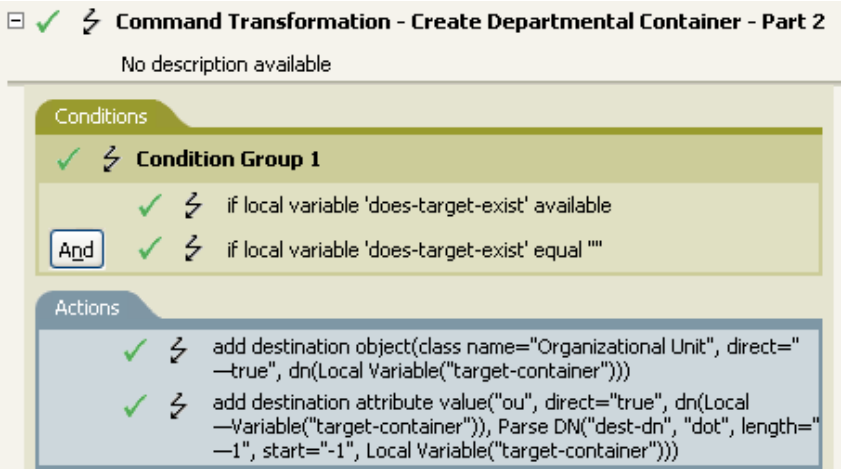
Remarks

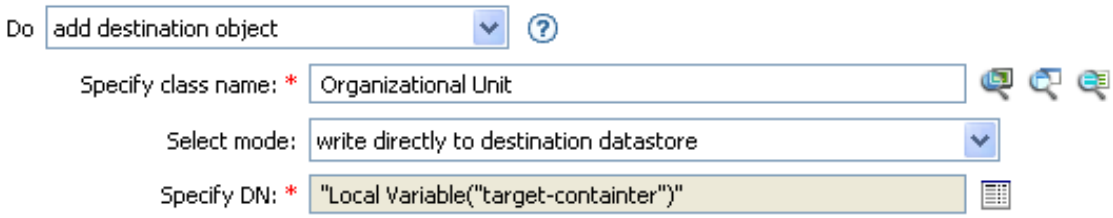
Any attribute values to be added as part of the object creation must be done in subsequent Add Destination Attribute Value actions, using the same DN.

Example

The example creates the department container that is needed. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Create Departmental Container - Part 1 and Part 2" on page 108 from the predefined rules. To see the policy in XML, see `predef_command_create_dept_container1.xml` (../samples/predef_command_create_dept_container1.xml) and `predef_command_create_dept_container2.xml` (../samples/predef_command_create_dept_container2.xml).







The OU object is created. The value for the OU attribute is created from the destination attribute value action that occurs after this action.

Add Resource

Initiates a request to Roles Based Provisioning Module (RBPM) for assigning the resource to the identity specified in the Resource DN field. The target identity is specified in the Object field. This action is available only with the OpenText Identity Manager server version 4.0.2 and later. If a policy containing this action encounters an error, OpenText Identity Manager generates an error in the `error.do-add-resource` local variable. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Resource DN

Specify the name of the resource to assign, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

Strings

(Optional) Specify additional argument strings for the Resource assignment request. You can enter the strings manually, or select the **Edit the Strings** icon. to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Add Resource action supports the following string arguments:

String Name	Description
description	A description of the reason for the request used for auditing and (if necessary) approval purposes. Default: Request generated by the policy.
CorrelationID	An identifier to correlate the role assignment process. Default: Operation event Correlation ID If no value is specified for the argument, it uses the default value. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.

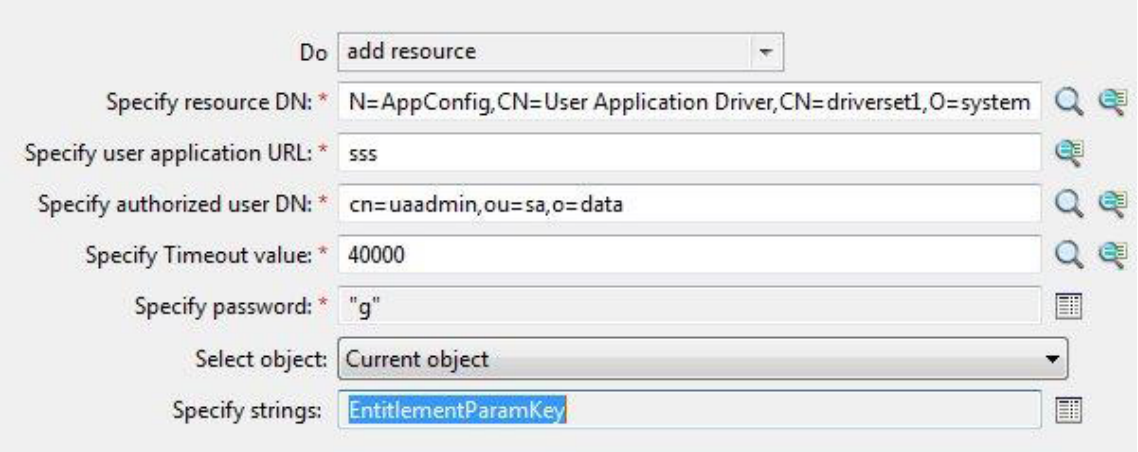
NOTE: You can specify parameter values for the added resources. You can use the plus sign (+) to insert a new string, or select the **Edit the Strings** icon to open the String Builder and specify the strings.

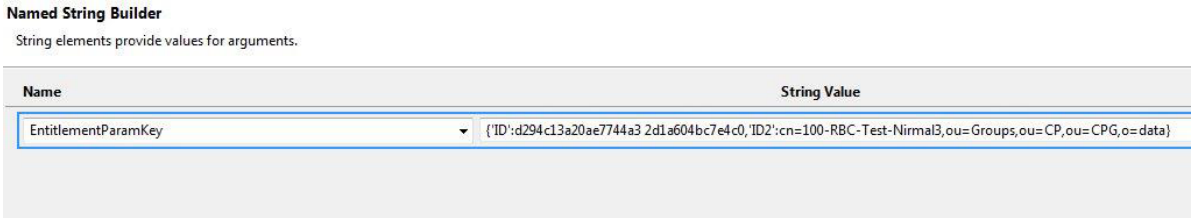
You must specify the parameter names as `param1`, `param2`, and so on.

If you add a dynamic resource, you must specify the parameter name as `EntitlementParamKey` and provide the value of the parameter in JSON format (for Identity Manager 4.0 and later) or the legacy entitlement format (for earlier versions of Identity Manager).

For more information about the Named String Builder, see "String Builder" on page 58.

Example





Add Role

Initiates a request to Roles Based Provisioning Module (RBPM) for assigning a role in the Role DN field to a user specified in the Object field. This action is available only with the OpenText Identity Manager server 3.6 or later. If a policy containing this action encounters an error, OpenText Identity Manager generates an error message in the `error.do-add-role` local variable. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Role DN

Specify the name of the role to assign, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the role assignment, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is `0`.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

Strings

(Optional) Specify additional argument strings for the Role assignment request. You can enter the strings manually, or select the **Edit the Strings** icon. to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Add Role action supports the following string arguments:

String Name	Description
role-assignment-type	The type of the role assignment. You can choose from the following options: ◆ USER_TO_ROLE ◆ GROUP_TO_ROLE ◆ CONTAINER_TO_ROLE ◆ ROLE_TO_ROLE Default: USER_TO_ROLE.
description	A description of the reason for the request used for auditing and (if necessary) approval purposes. Default: Request generated by the policy.
effective-time	The time (in CTIME format) the role assignment should become effective. Default: now
expiration-time	The time (in CTIME format) the role assignment automatically expires. Default: never
sod-justification	A justification for requesting an exception for any Separation of Duty violations this assignment will trigger. Default: No exception will be requested and the request will fail if it causes a violation. **NOTE:** By default, the Named String Builder does not display this string. However, you can manually add it to the string list.
CorrelationID	An identifier to correlate the role assignment process. Default: Operation event Correlation ID If no value is specified for the argument, it uses the default value. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.
originator	The originator of the role assignment request.

Add Source Attribute Value

Adds the specified attribute on an object in the source data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN

Specify the DN, association, or the current object as the target object.

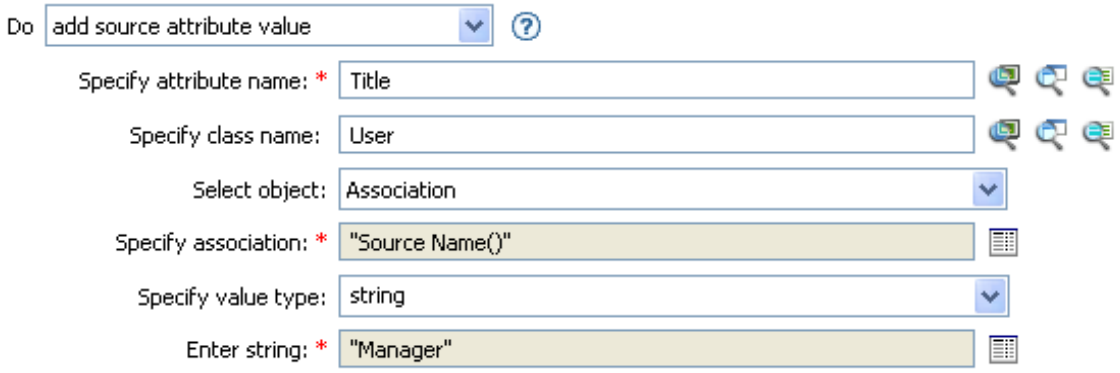
Value Type

Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

String

Specify the attribute value to be added.

Example



Add Source Object

Creates an object of the specified type in the source data store, with the name and location provided in the DN field. Any attribute values to be added as part of the object creation must be done in subsequent Add Source Attribute Value actions, using the same DN.

Fields

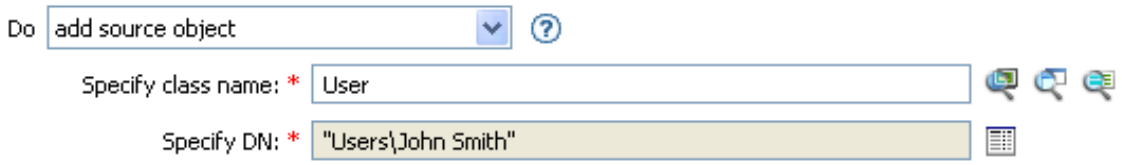
Class Name

Specify the class name of the object to be added. Supports variable expansion. For more information, see "Variable Selector" on page 31.

DN

Specify the DN of the object to be added.

Example



Append XML Element

Appends a custom element, with the name specified in the **Name** field, to the set of elements selected by the XPath expression. If **Before XPath Expression** is not specified, the new element is appended after any existing children of the selected elements. If **Before XPath Expression** is specified, it is evaluated relative to each of the elements selected by the expression to determine which of the children to insert before. If **Before XPath Expression** evaluates to an empty node set or a node set that does not contain any children of the selected element, the new element is appended after any existing children; otherwise, the new element is inserted before each of the nodes in the node set selected by before that are children of the selected node.

Fields

Name

Specify the tag name of the XML element. This name can contain a namespace prefix if the prefix has been previously defined in this policy. Supports variable expansion. For more information, see "Variable Selector" on page 31.

XPath Expression

Specify an XPath 1.0 expression that returns a node set containing the elements to which the new elements should be appended.

Insert

Select whether to insert the XPath expression before the source XPath expression or append the XPath expression to the end of the current node in the destination XPath expression.

Before XPath Expression

Specify an XPath 1.0 expression that evaluates relative to each of the nodes selected by the expression that returns a node set containing the child nodes that the new elements should be inserted before. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example

If you are implementing Credential Provisioning policies, there is a sample Subscriber Command Transformation policy that uses the XPath Expression condition. The sample file is called `SampleSubCommandTransform.xml`. It is found in the `\dirxml\utilities` folder on the OpenText Identity Manager media. For more information, see "Example Credential Provisioning Policies" in *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*.

The sample file uses the append XML element action to add the SecureLogin or SecretStore credentials to the user object when it is provisioned.

Add operation-data element to password subscribe operations (if needed)

Add payload data to modify-password subscribe operations

Add payload data to add subscribe operations

No description available

Conditions

Condition Group 1

if operation equal "add"

And

if password available

Actions

```
append XML element("sso-sync-data", "operation-data")
append XML element("sso-target-user-dn", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-target-user-dn", Source Attribute("DirXML-ADContext"))
append XML element("sso-app-username", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-app-username", Source Attribute("CN"))
append XML element("password", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/password", Password())
append XML element("nsl-set-passphrase-answer", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/nsl-set-passphrase-answer", Source Attribute("workforceID"))
```

Do: append XML element

Enter element name: * sso-sync-data

Specify XPath expression: * operation-data

Insert: Append to end of XPath expression

Append XML Text

Appends the specified text to the set of elements selected by the XPath expression. If Before XPath Expression is not specified, the text is appended after any existing children of the selected elements. If Before XPath Expression is specified, it is evaluated relative to each of the elements selected by the expression to determine which of the children to insert before. If Before XPath Expression evaluates to an empty node set or a node set that does not contain any children of the selected element, then the text is appended after any existing children; otherwise, the text is inserted before each of the nodes in the previously selected node set that are children of the selected node.

Fields

XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the elements to which the new elements should be appended. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

String

Specify the text to be appended.

Insert

Select whether to insert the XPath expression before the source XPath expression or append the XPath expression to the end of the current node in the destination XPath expression.

Before XPath Expression

Specify the XPath 1.0 expression that evaluates relative to each of the nodes selected by the expression that returns a node set containing the child nodes that the text should be inserted before. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example

If you are implementing Credential Provisioning policies, there is a sample Subscriber Command Transformation policy that uses the XPath Expression condition. The sample file is called `SampleSubCommandTransform.xml`. It is found in the `\dirxml\utilities` folder on the OpenText Identity Manager media. For more information, see "Example Credential Provisioning Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. To view the policy in XML, see `SampleSubCommandTransform.xml` (../samples/SampleSubCommandTransform.xml).

The example is using the append XML text action to find the SecureLogin or SecretStore application username. By obtaining the application name, the credentials can be set for the user object when it is provisioned.

Add operation-data element to password subscribe operations (if needed)

Add payload data to modify-password subscribe operations

Add payload data to add subscribe operations

No description available

Conditions

Condition Group 1

if operation equal "add"

And

if password available

Actions

```
append XML element("sso-sync-data", "operation-data")
append XML element("sso-target-user-dn", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-target-user-dn", Source Attribute("DirXML-ADContext"))
append XML element("sso-app-username", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/sso-app-username", Source Attribute("CN"))
append XML element("password", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/password", Password())
append XML element("nsl-set-passphrase-answer", "operation-data/sso-sync-data")
append XML text("operation-data/sso-sync-data/nsl-set-passphrase-answer", Source Attribute("workforceID"))
```

Do: append XML text

Specify XPath expression: * operation-data/sso-sync-data/sso-target-user-dn

Specify string: * "Source Attribute("DirXML-ADContext")"

Insert: Append to end of XPath expression

Break

Ends processing of the current operation by the current policy.

Example

Clear Destination Attribute Value

Removes all values for the named attribute from an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

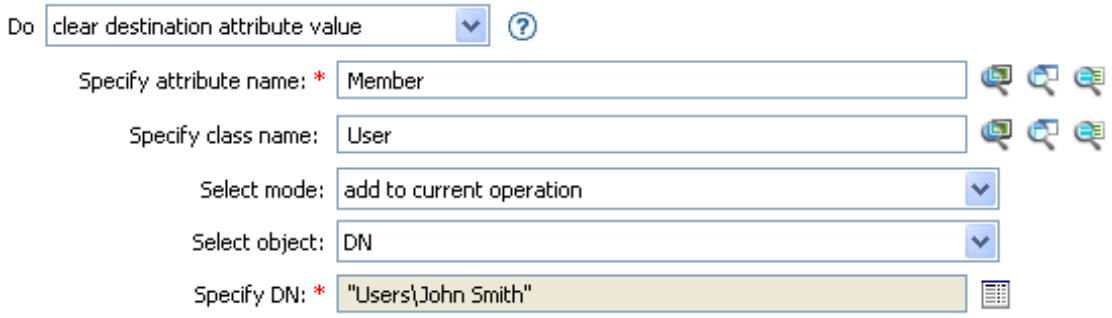
Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN

Select the DN, association, or current object as the target object.

Example



Clear Operation Property

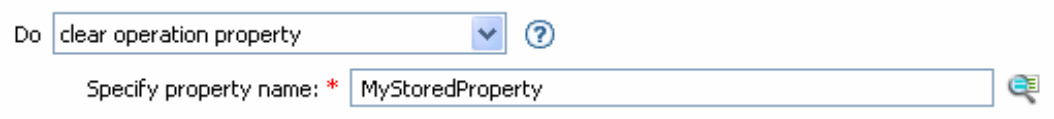
Clears any operation property with the provided name from the current operation. The operation property is the XML attribute attached to an `<operation-data>` element by a policy. An XML attribute is a name/value pair associated with an element in the XDS document.

Fields

Property Name

Specify the name of the operation property to clear. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example



Clear Source Attribute Value

Removes all values of an attribute from an object in the source data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. This value might be required for schema map purposes if the object is other than current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

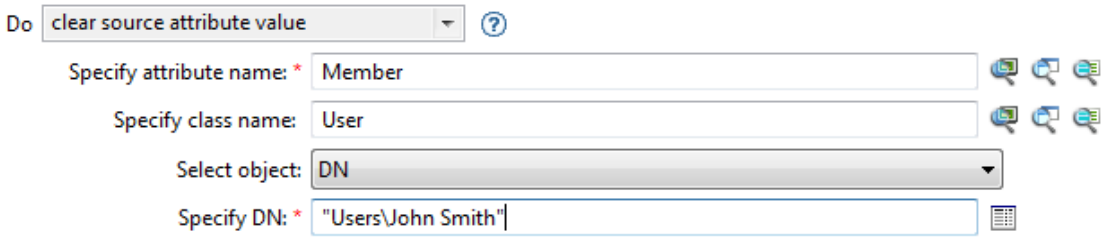
Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN

Select the DN, association, or current object as the target object.

Example



Clear SSO Credential

Clears the Single Sign On credential so objects can be deprovisioned. Additional information about the credential to be cleared can be provided in the **Enter login parameter strings** field. The number of the strings and the names used are dependent on the credential repository and application for which the credential is targeted. For more information, see the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-clear-sso-credential`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Target User DN

Specify the DN of the target users.

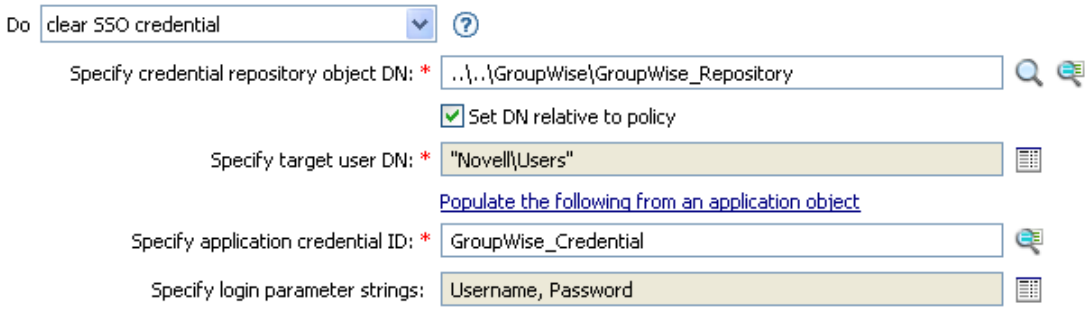
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object. You can enter the strings manually, or select the **Edit the Strings** icon to open the String Builder and specify the strings. For more information about the Named String Builder, see "String Builder" on page 58.

Example



Clone By XPath Expressions

Appends deep copies of the nodes specified by the source field to the set of elements specified by the destination field. If Before XPath Expression is not specified, the non-attribute cloned nodes are appended after any existing children of the selected elements. If Before XPath Expression is specified, it is evaluated relative to each of the elements selected by expression to determine which of the children to insert before. If Before XPath Expression evaluates to an empty node set or a node set that does not contain any children of the selected element, the non-attribute cloned nodes are appended after any existing children; otherwise, the non-attribute cloned nodes are inserted before each of the nodes in the previously selected node set that are children of the selected node.

Fields

Source XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the nodes to be copied. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

Destination XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the elements to which the copied nodes are to be appended. Supports variable expansion. For more information on variable expansion and XPath, see "Variable Selector" on page 31.

Insert

Select whether to insert the XPath expression before the source XPath expression or append the XPath expression to the end of the current node in the destination XPath expression.

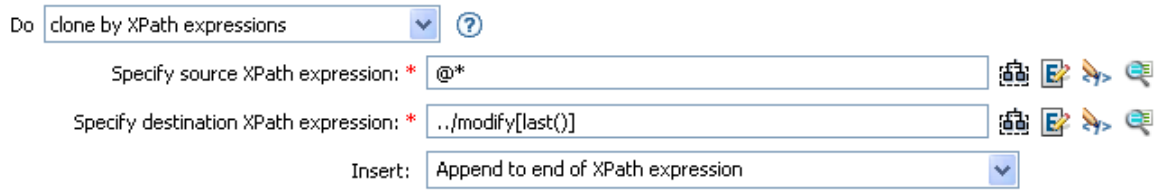
Before XPath Expression

Specify the XPath 1.0 expression that evaluates relative to each of the nodes selected by the expression that returns a node set containing the child nodes that the text should be inserted before. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example



Clone Operation Attribute

Copies all occurrences of an attribute within the current operation to a different attribute within the current operation.

Fields

Source Name

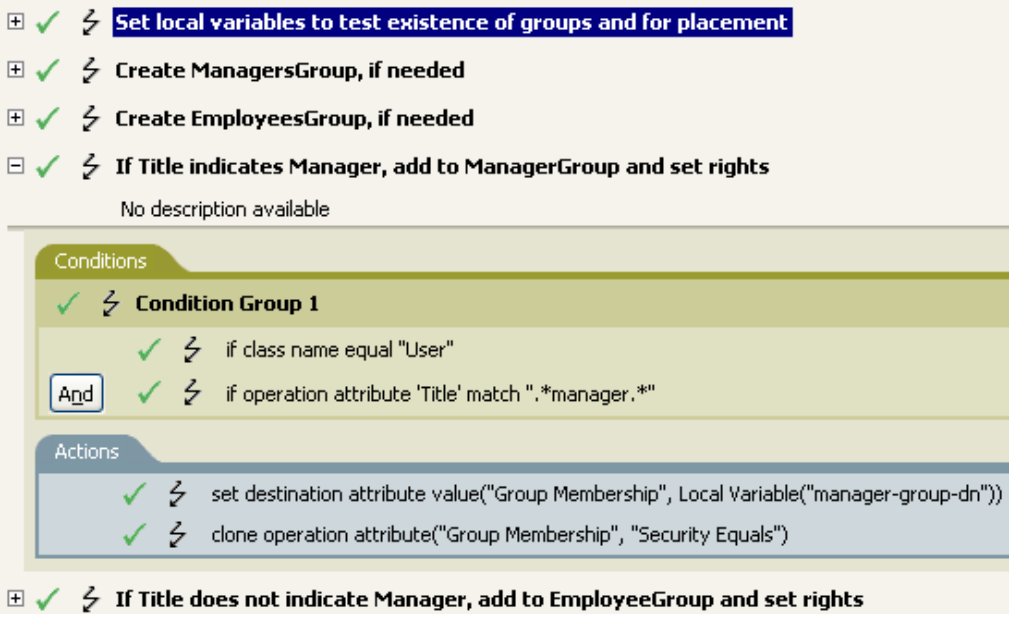
Specify the name of the attribute to be copied from. Supports variable expansion. For more information, see "Variable Selector" on page 31.

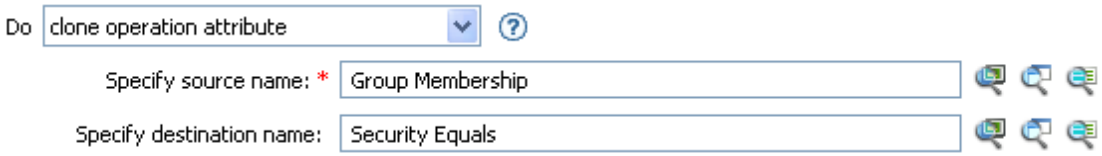
Destination Name

Specify the name of the attribute to be copied to. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

The example adds a User object to the appropriate Employee or Manager group based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy is Govern Groups for User Based on Title Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To see the policy in XML, see `003-Command-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).





The Clone Operation Attribute is taking the information from the Group Membership attribute and adding it to the Security Equals attribute so the values are the same.

Create Resource

Initiates a request to Roles Based Provisioning Module (RBPM) for creating a resource specified in the **Resource Name** field. If the distinguished name of the entitlement is specified in the request, OpenText Identity Manager creates a resource with an entitlement. Otherwise, the resource is created without an entitlement. To request for a static resource, specify the value of the entitlement. Specifying an entitlement value is not needed for creating a dynamic resource. When the action is successfully performed, OpenText Identity Manager generates a success message in the `success.do-create-resource` local variable. If a policy containing this action encounters an error, an error message is generated in the `error.do-create-resource` local variable. For more information about local variables, see "Local Variable Selector" on page 67. This action is available only with the OpenText Identity Manager server version 4.6 and later.

Fields

Resource Name

Specify the name of the resource to create. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

REST WebService

NOTE: This option is introduced in Identity Manager 4.8.2. If you are on a prior version of Identity Manager, you can use the REST APIs by directly editing the XML file for the do-create-resource action. For more information, see the OpenText Identity Manager Designer Administration Guide.

Specify whether you want to use REST API. To use REST API, set the value to **true**. The default value is **false**.

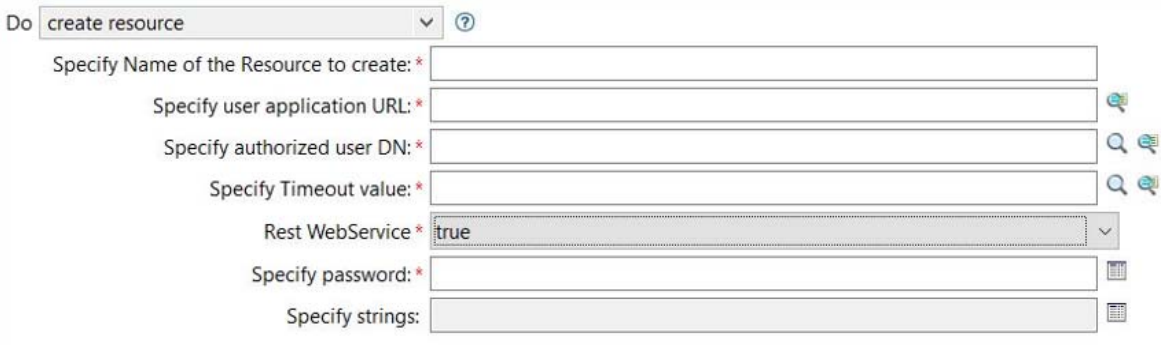
Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

Strings

(Optional) Specify additional argument strings for the Resource creation request. You can enter the strings manually or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Example



The Create Resource action supports the following string arguments:

String Name	Description
Description	A description of the reason for the request used for auditing and approval purposes if necessary. Default: Request generated by policy.
Display Name	Display name of the resource to be created. Default: Resource name.
Entitlement DN	DN of the entitlement in LDAP format.
Static	The type of resource to be created. You must specify the entitlement value while creating a static resource.

Entitlement Value	The value of the entitlement in JSON format. This is only needed if you are creating a static resource.
	
	For example,
	
	`{`
	`"ID": "f1e84f2a7964614eaa45407c724e3a98",`
	`"ID2": "CN=Domain Users,CN=Users,DC=yourcompany,DC=msft"`
	
	`}`

Category Key	The category in which the resource should be created. For example, system, default, or both.
Owner	The owner of the resource in LDAP format. Multiple owners are allowed for a resource. Specify multiple owners in a semi colon(;) separated list.
Grant Approver	The approver of the resource assignment in LDAP format. Multiple approvers are allowed. Specify multiple approvers in a semi colon(;) separated list to form a serial approval process.
Grant Quorum	Minimum percentage of approvals required for creating a resource.

String Name	Description
Revoke Approver	Approver who has the rights for revoking a resource in LDAP format. Leave this field blank if this is the same approver who granted the resource. Multiple approvers are allowed to revoke a resource. Specify multiple approvers in a semi colon(;) separated list, which forms a serial approval process.
Revoke Quorum	Minimum percentage of approval required for revoking a resource.
Allow Override	Specifies whether the role approval overrides the resource approval. Default: false
Multi-valued	Specifies whether the resource has multiple entitlement values. Default: false
PRD DN	DN of Provisioning Request Definition in LDAP format.
SubContainer	Indicates the LDAP DN of the container in which the resource will be created. This applies only when the REST API is used.

Create Role

Initiates a request to Roles Based Provisioning Module (RBPM) for creating a role specified in the **Role Name** field. When the action is successfully performed, OpenText Identity Manager generates a success message in the `success.do-create-role` local variable. If a policy containing this action encounters an error, Identity Manger generates an error message in the `error.do-create-role` local variable. For more information about local variables, see "Local Variable Selector" on page 67. This action is available only with the OpenText Identity Manager server version 4.6 and later.

Fields

Role Name

Specify the name of the role to create. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

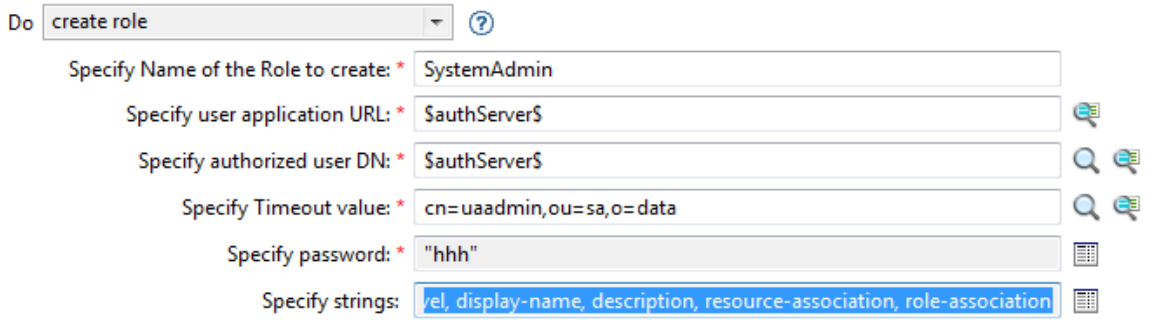
Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Strings

(Optional) Specify additional argument strings for the Resource assignment request. You can enter the strings manually, or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Example



The Create Role action supports the following string arguments:

String Name	Description
Role Level	The level of the role. For example, 10, 20 or 30. Default: 10
Display Name	Display name of the role be created. Default: Role name.
Description	A description of the reason for the request used for auditing and approval purposes if necessary. Default: Request generated by the policy.
Category Key	The category in which the role should be created. For example, system, default, or both.
Owner	The owner of the role in LDAP format. Multiple owners are allowed for a resource. Specify multiple owners in a semi colon(;) separated list.
Grant Approver	The approver of the role assignment in LDAP format. Multiple approvers are allowed. Specify multiple approvers in a semi colon(;) separated list to form a serial approval process.
Grant Quorum	Minimum percentage of approvals required for creating a role.
Subcontainer	Directory container under the role level for storing the role. For example, `cn=netiq`, where `netiq` specifies the name of the subcontainer.
Resource Association	Resource association for the role. This element should have a resource name, resource association description, and the entitlement value separated by a semi colon(;). *Resource DN in LDAP format;Resource association description;Entitlement Value.* You can add multiple **resource-association** elements to associate multiple resources with a role. For creating a static resource, entitlement value is not needed.
Role Association	Role association for the role. This element should have a role name, role assignment description, and the role relationship separated by semi colon(;). *Role DN in LDAP format;Role assignment description;Relationship.* The role relationship can be a child or a parent. You can add multiple **role-association** elements to assign multiple roles with a role.
Revoke Approver	Approver who has the rights for revoking a resource in LDAP format. Leave this field blank if this is the same approver who granted the resource. Multiple approvers are allowed to revoke a resource. Specify multiple approvers in a semi colon(;) separated list, which forms a serial approval process.

Delete Destination Object

Deletes an object in the destination data store.

Fields

Class Name

(Optional) Specify the class name of the object to delete in the destination data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

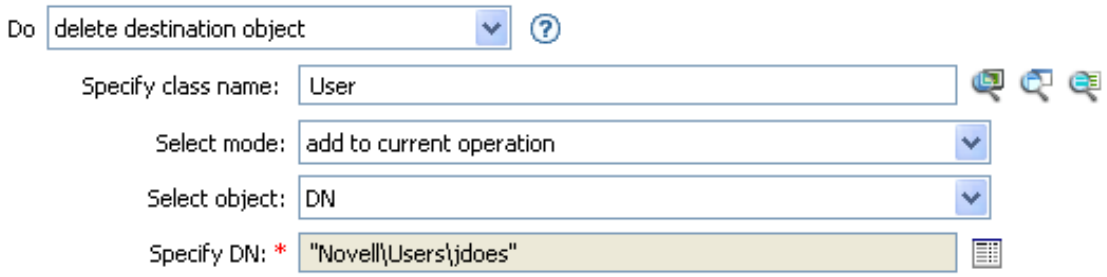
Object

Select the target object type to delete in the destination data store. This object can be the current object, or can be specified by a DN or an association.

DN

Select the DN, association, or current object as the target object.

Example



Delete Source Object

Deletes an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to delete in the source data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

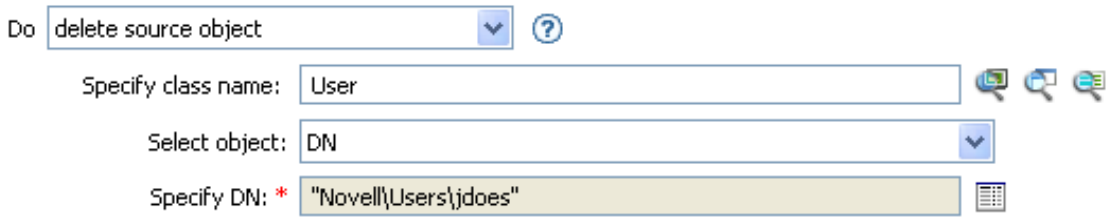
Object

Select the target object type to delete in the source data store. This object can be the current object, or can be specified by a DN or an association.

DN

Select the DN, association, or current object as the target object.

Example



Delete Resource

Initiates a request to the Roles Based Provisioning Module (RBPM) for deleting a resource specified in the **Resource Name** field. The credentials provided in the ID and first `<arg-password>` are used for placing the request to the Identity Applications server. It uses the OpenText Identity Manager REST APIs which internally uses the OAuth2 protocol for authentication. The OSP client ID (`osp-clientid`) should be specified for authentication. The client password should be specified by the second `<arg-password>`. You can specify the additional optional arguments to the Resource creation request through the named `<arg-string>'s`.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Resource Name

Description of the resource name.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

OSP Client ID

Specify the client ID to authenticate to OSP. Supports variable expansion.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. Format of the URL must be: https://[Identity Applications Server DNS or IP Address]:[Port]. The https port is 8543 by default. For more information, see "Variable Selector" on page 31 and Identity Applications REST API Documentation (https://www.netiq.com/documentation/identity-manager-developer/rest-api-documentation/idmappsdoc/#/Catalog) page.

Strings

(Optional) Specify additional argument strings for the Resource creation request. You can enter the strings manually or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Delete Role

Initiates a request to the Roles Based Provisioning Module (RBPM) for deleting a role specified in the **Role Name** field. The credentials provided in the ID and first `<arg-password>` are used for placing the request to the Identity Applications server. It uses the OpenText Identity Manager REST APIs which internally uses the OAuth2 protocol for authentication. The OSP client ID (`osp-clientid`) should be specified for authentication. The client password should be specified by the second `<arg-password>`. You can specify the additional optional arguments to the Resource creation request through the named `<arg-string>'s.`

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Role Name

Description of the role name.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

OSP Client ID

Specify the client ID to authenticate to OSP. Supports variable expansion.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Strings

(Optional) Specify additional argument strings for the Resource creation request. You can enter the strings manually or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Find Matching Object

Finds a match for the current object in the destination data store. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-find-matching-object`. The `error.do-find-matching-object` stores the error and is of local variable scope. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Scope

Select the scope of the search. The scope might be an entry, a subordinate, or a subtree.

DN

Specify the DN that is the base of the search.

Match Attributes

Specify the attribute values to search for.

IMPORTANT: To improve performance when using the find matching object verb, create an index for the attributes that you are going to use when querying the Identity Vault. For more information about indexes, see the OpenText eDirectory Administration Guide (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/a5tuuu5.html).

Remarks

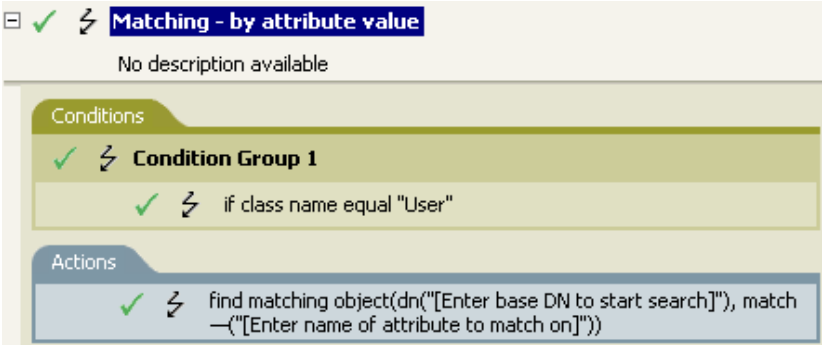
Find Matching Object is only valid when the current operation is an add.

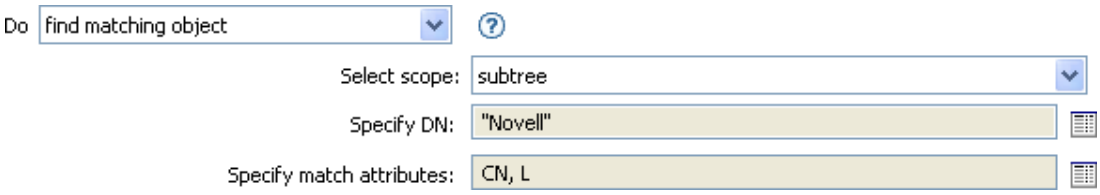
The DN argument is required when the scope is "entry," and is optional otherwise. At least one match attribute is required when the scope is "subtree" or "subordinates."The results are undefined if the scope is "entry" and there are match attributes specified. If the destination data store is the connected application, then an association is added to the current operation for each successful match that is returned. No query is performed if the current operation already has a non-empty association, thus allowing multiple find matching object actions to be strung together in the same rule. If more than one result is returned, then the local variable error.do-find-matching-object will be set to a node-set containing the list of source DNs from the instances if they are available, or the list of associations if the source DN's are not available.

If the destination data store is the Identity Vault, then the destination DN attribute for the current operation is set. No query is performed if the current operation already has a non-empty destination DN attribute, thus allowing multiple find matching object actions to be strung together in the same rule. If only a single result is returned and it is not already associated, then the destination DN of the current operation is set to the source DN of the matching object. If only a single result is returned and it is already associated, then the destination DN of the current operation is set to the single character ￼ and the local variable error.do-find-matching-object is set to the source DN from that matching object. If multiple results are returned, then the destination DN of the current operation is set to the single character &#xFFFD; and the local variable error.do-find-matching-object is set to a node-set containing the source DN's from those matching objects.

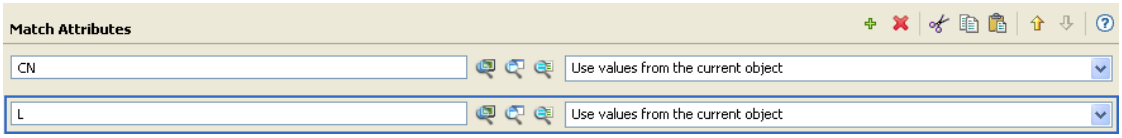
Example

The example matches on User objects with the attributes CN and L. The location where the rule is searching starts at the Users container and adds the information stored in the OU attribute to the DN. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Matching - By Attribute Value" on page 129. To see the policy in XML, see `predef_match_by_attribute.xml` (../samples/predef_match_by_attribute.xml).





When you click the **Argument Builder** icon, the Match Attribute Builder comes up. You specify the attribute you want to match on in the builder. This example uses the CN and L attributes.

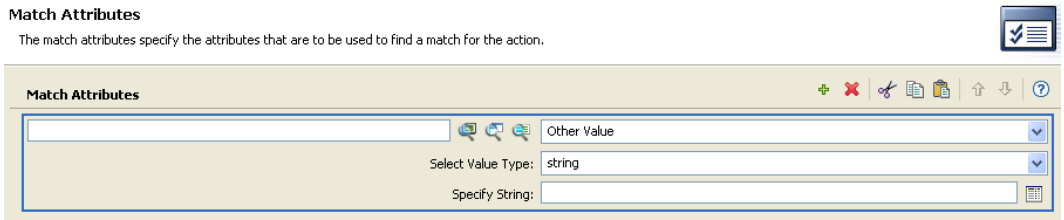


The left fields store the attributes to match. The right fields allow you to specify to use the value from the current object to match or to use another value. If you select **Other Value**, there are multiple value types to specify:

- counter
- dn
- int
- interval
- octet
- state
- string
- structured
- teleNumber
- time

To use another value:

1 Launch the Match Attribute Builder by selecting **Edit the match attributes**, then select **Other Value**.



2 Select the desired value type.

3 Specify the value, then click **Finish**.

For Each

Repeats a set of actions for each node in a node set.

Fields

Node Set

Specify the node set.

Action

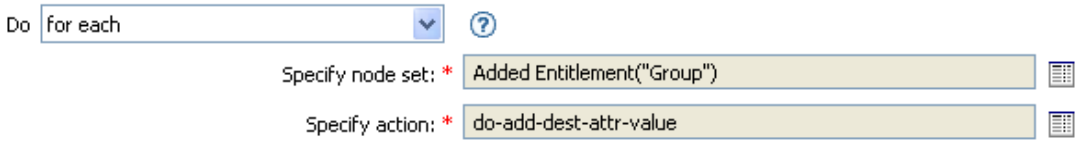
Specify the actions to perform on each node in the node set.

Remarks

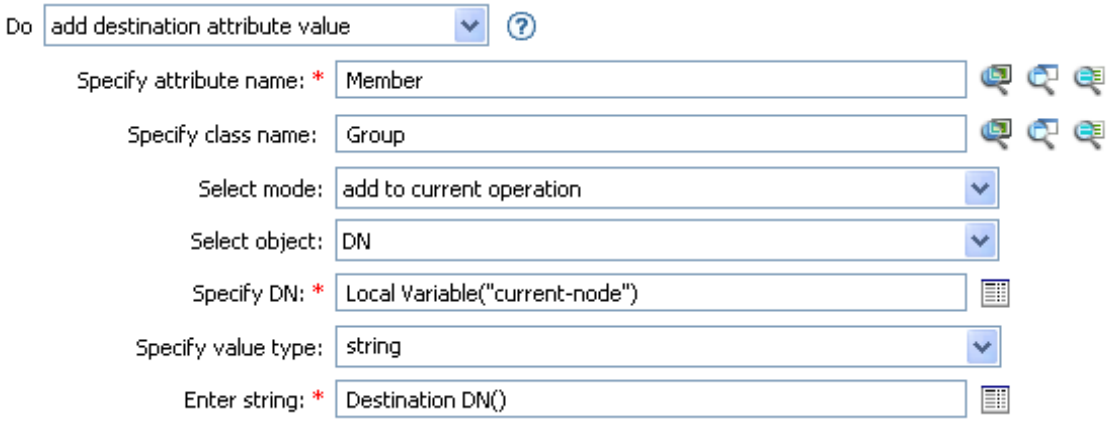
The current node is a different value for each iteration of the actions, if a local variable is used.

If the current node in the node set is an entitlement element, then the actions are marked as if they are also enclosed in an Implement Entitlement action. If the current node is a query element returned by a query, then that token is used to automatically retrieve and process the next batch of query results.

Example



The following is an example of the Actions Builder, used to provide the action argument:



For more information on the Action Argument Component Builder, see "Actions Builder" on page 42.

Generate Event

Sends a user-defined event to Sentinel.

Fields

ID

ID of the event. The provided value must result in an integer in the range of 1000-1999 when parsed by using the parseInt method of java.lang.Integer. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Level

Level of the event.

Level	Description
log-informational	Positive events of any importance.
log-alert	Events that require immediate attention.
log-critical	Events that can cause parts of the OpenText Identity Manager engine or driver to malfunction.
log-debug	Events of relevance for support or engineers to debug the operation of the OpenText Identity Manager engine or driver.
log-emergency	Events that cause the OpenText Identity Manager engine or driver to shut down.
log-error	Events describing errors that can be handled by the OpenText Identity Manager engine or driver.
log-notice	Events (positive or negative) that an administrator can use to understand or improve use and operation.
log-warning	Negative events not representing a problem.

Strings

Specify user-defined string, integer, and binary values to include with the event. You can enter the strings manually, or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Generate Event action supports the following strings:

String Name	Description
data	Data entered here is stored in the blob event field.
data-type	Specifies the data-type of the value in the data tag.
subTarget	The subcomponent of the target being acted upon.
target	The object being acted upon.

String Name	Description
target-type	Integer specifying a predefined format for the target. Predefined values for target-type are currently: ◆ 0 = None ◆ 1 = Slash Notation ◆ 2 = Dot Notation ◆ 3 = LDAP Notation
text1	Text entered here is stored in the text1 event field.
text2	Text entered here is stored in the text2 event field.
text3	Text entered here is stored in the text3 event field.
value	Any number entered here is stored in the value event field. You can also access this field using the `value1` tag.
value3	Any number entered here is stored in the value3 event field.

Remarks

The Sentinel event structure contains a target, a subTarget, three strings (text1, text2, text3), two integers (value, value3), and a generic field (data). The text fields are limited to 256 bytes, and the data field can contain up to 3 KB of information, unless a larger data field is enabled in your environment.

Example

The example has four rules that implement a placement policy for User objects based on the first character of the Surname attribute. It generates both a trace message and a custom Sentinel event. The Generate Event action is used to send Sentinel an event. The policy name is Policy to Place by Surname and is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Placement-BySurname.xml` (../samples/001-Placement-BySurname.xml).

Setup Local Variables

Surname A-I: place in Users1

No description available

Conditions

Condition Group 1

if class name equal "User"

And if operation attribute 'Surname' match "[a-i].*"

Actions

set operation destination DN(dn("Training\Users\Active\Users1"+"\"+Operation Attribute("CN")))

trace message(color="yellow", Local Variable("LVUsers1"))

generate event(id="1000", text1=Local Variable("LVUsers1"))

Surname J-R: place in Users2

Surname S-Z: place in Users3

Do generate event

Specify ID: * 1000

Select level: informational

Specify strings: text1

If

Conditionally performs a set of actions.

Fields

If Conditions

Specify the desired condition.

Then Perform Actions

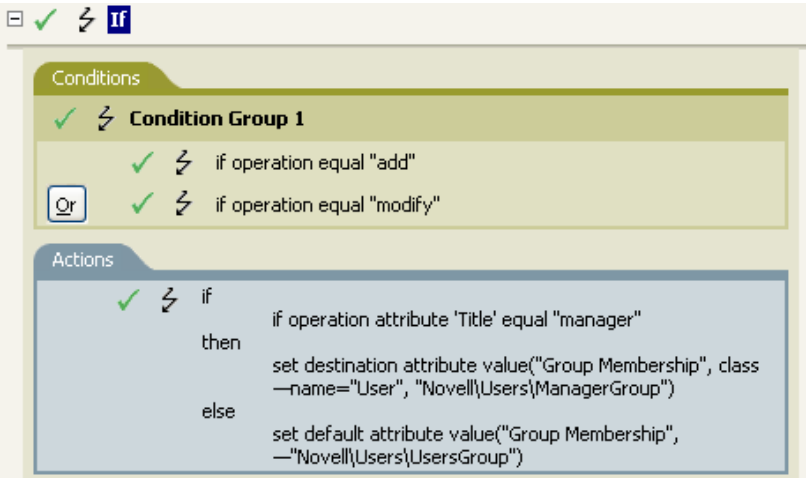
Specify the desired actions, if the conditions are True.

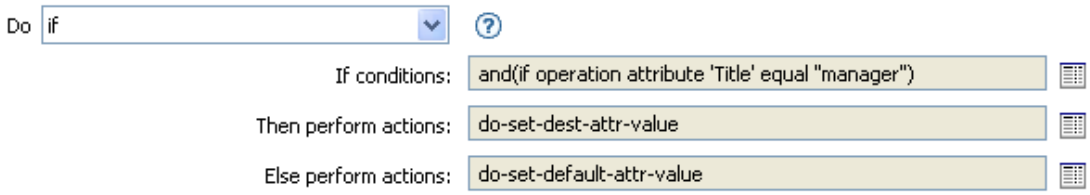
Else Perform Actions

(Optional) Specify the desired actions, if the conditions are False.

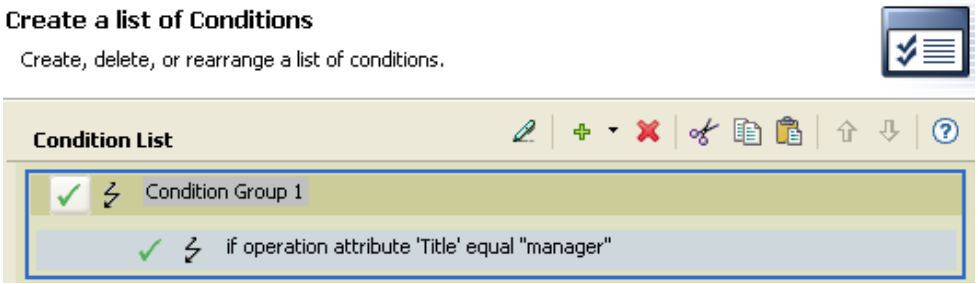
Example

During an Add or Modify operation, if the attribute of Title equals manager, the user object is added to the ManagerGroup group. If the Title does not equal manager, then the user object is added to the UsersGroup group. To view the policy in XML, see `if.xml` (../samples/if.xml).



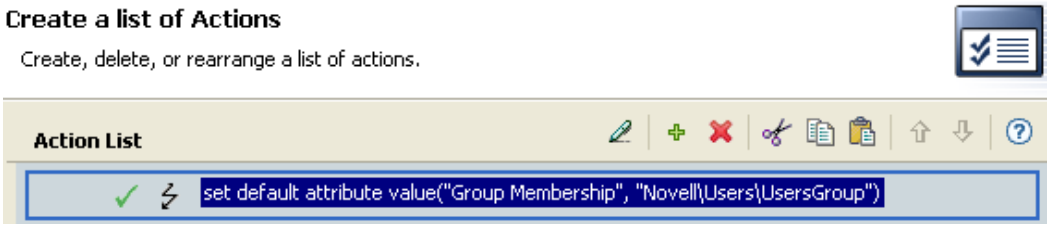


When you create the if action, you must add a condition and one action. In this example, there are two separate actions. The condition is if a user object has the title of manager.



The action is to add the user object to the ManagerGroup group.

If the title does not equal manager, the user object is placed in the UsersGroup group.



If Operation Entitlement

Conditionally performs a test on entitlements of the current object in the current operation.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

Name

Specify the name of the entitlement. Supports variable expansion.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Example

The If Operation Entitlement action supports the following string arguments:

String Name	Description
Available	The named entitlement is available and granted in the current operation.
Changing	The current operation contains a change (grant or revoke) of the named entitlement.
Changing From	The current operation contains a change that revokes a value of the named entitlement that has a value that equals the content of the condition when compared using the specified comparison mode. Supports variable expansion.
Changing To	The current operation contains a change that grants a value of the named entitlement that has a value that equals the content of the condition when compared using the specified comparison mode. Supports variable expansion.
Equal	The named entitlement is available and granted in the current operation and has a value that equals the content of the condition when compared using the specified comparison mode.Resource Name in LDAP format;Resource association description;Entitlement Value. Supports variable expansion.
lt	The named entitlement is available and granted in the current operation and has a value that is less than the content of the condition when compared using the specified comparison mode. Supports variable expansion.

String Name	Description
gt	The named entitlement is available and granted in the current operation and has a value that is greater than the content of the condition when compared using the specified comparison mode. Supports variable expansion.
Not Available	This value returns true when the `available` value returns false.
Not Changing	This value returns true when the `changing` value returns false.
Not Changing From	This value returns true when the `changing from` value returns false.
Not Changing To	This value returns true when the `changing to` value returns false.
Not Equal	This value returns true when the `equal` value returns false.
Not lt	This value returns true when the `lt` value returns false.
Not gt	This value returns true when the `gt` value returns false.

Implement Entitlement

Designates actions that implement an entitlement so that the status of those entitlements can be reported to the agent that granted or revoked the entitlement.

Fields

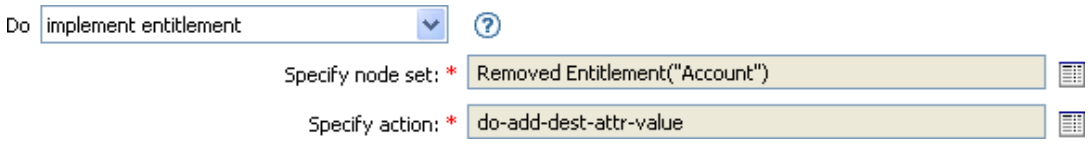
Node Set

Node set containing the entitlement being implemented by the specified actions.

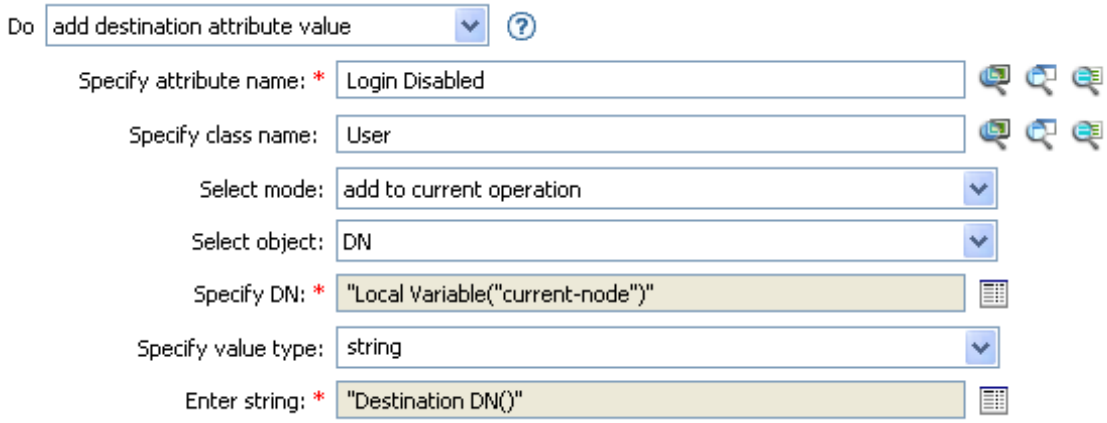
Action

Actions that implement the specified entitlements.

Example



The following is an example of the Actions Builder, used to provide the action argument:



For more information on the Actions Builder, see "Actions Builder" on page 42.

Invoke REST Endpoint

Initiates a REST request to the endpoint specified by URL using the authentication method specified by `auth-method`. The type of the request (POST, PUT, GET, DELETE) should be specified by the `type` identifier. If the authentication method is `oauth2`, then the authentication URL is specified by `auth-url`. The request is made using credentials specified by ID and `<arg-password>`. Additional optional arguments to the REST request may be specified by named `<arg-string>'s`. There will be `success.do-invoke-rest-endpoint` or `error.do-invoke-rest-endpoint` local variable available to the enclosing policy depending on the success or failure of this request and responses get saved in the respective variable. The `success.do-invoke-rest-endpoint` local variable will be available only if the endpoint is invoked successfully. The `error.do-invoke-rest-endpoint` will be available only if any type of error occurs while invoking the endpoint and it contains the error string. There are no retry options available.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Authentication Method

Specify the authentication method you want to use. The supported values are `basic` or `oauth2`. Supports variable expansion.

Authentication URL

Specify the authentication URL. You must specify the URL only when the authentication method is oauth2. Supports variable expansion.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Example

The Invoke REST Endpoint action supports the following string arguments:

String Name	Description
Payload	Specifies the payload for the REST request.
Truststore	Specifies the trust store name.
Truststore Password	Specifies the trust store password
Authentication Query Name	Specifies the authentication query name.
Authentication Query Value	Specifies the authentication query value.
Authentication Header Name	Specifies the authentication header name.
Authentication Header Value	Specifies the authentication header value.
Operation Header Name	Specifies the operation header name.
Operation Header Value	Specifies the operation header value.

Modify Resource

Initiates a request to the Roles Based Provisioning Module (RBPM) for modifying a Resource specified in the **Resource Name** field. The credentials provided in the ID and first `<arg-password>` are used for placing the request to the Identity Applications server. It uses the OpenText Identity Manager REST APIs which internally uses the OAuth2 protocol for authentication. The OSP client ID (`osp-clientid`) should be specified for authentication. The client password should be specified by the second `<arg-password>`. You can specify the additional optional arguments to the Resource creation request through the named `<arg-string>'s`.

Fields

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Resource Name

Description of the resource name.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

OSP Client ID

Specify the client ID to authenticate to OSP. Supports variable expansion.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Strings

(Optional) Specify additional argument strings for the Resource creation request. You can enter the strings manually or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Example

The Modify Resource action supports the following string arguments:

String Name	Description
Description	A description of the reason for the request used for auditing and approval purposes if necessary. Default: Request generated by policy

String Name	Description
Approval Required	Specifies whether an approval is required for this resource assignment Default: False
Entitlement DN	DN of the entitlement in LDAP format.
Revoke Definition	Fully Qualified LDAP DN of the Request definition for resource approval.
Revoke request Definition	Fully Qualified LDAP DN of the Request definition for revoking the resource.
Category Key	The category in which the resource should be created. For example, system, default, or both.
Owner	The owner of the resource in LDAP format. Multiple owners are allowed for a resource. Specify multiple owners in a semi colon(;) separated list.
Grant Approver	The approver of the resource assignment in LDAP format. Multiple approvers are allowed. Specify multiple approvers in a semi colon(;) separated list to form a serial approval process.
Grant Quorum	Minimum percentage of approvals required for modifying a resource.
Revoke Approver	Approver who has the rights for revoking a resource in LDAP format. Leave this field blank if this is the same approver who granted the resource. Multiple approvers are allowed to revoke a resource. Specify multiple approvers in a semi colon(;) separated list, which forms a serial approval process.
Revoke Quorum	Minimum percentage of approval required for revoking a resource.

Modify Role

Initiates a request to the Roles Based Provisioning Module (RBPM) for modifying a Role specified in the **Role Name** field. The credentials provided in the ID and first `<arg-password>` are used for placing the request to the Identity Applications server. It uses the OpenText Identity Manager REST APIs which internally uses the OAuth2 protocol for authentication. The OSP client ID (`osp-clientid`) should be specified for authentication. The client password should be specified by the second `<arg-password>`. You can specify the additional optional arguments to the Resource creation request through the named `<arg-string>'s`.

Fields

Role Name

Description of the role name.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

OSP Client ID

Specify the client ID to authenticate to OSP. Supports variable expansion.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Strings

(Optional) Specify additional argument strings for the Resource creation request. You can enter the strings manually or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Example

The Modify Role action supports the following string arguments:

String Name	Description
Category Key	The category in which the resource should be created. For example, system, default, or both.

String Name	Description
Owner	The owner of the resource in LDAP format. Multiple owners are allowed for a resource. Specify multiple owners in a semi colon(;) separated list.
Grant Approver	The approver of the role assignment in LDAP format. Multiple approvers are allowed. Specify multiple approvers in a semi colon(;) separated list to form a serial approval process.
Grant Quorum	Minimum percentage of approvals required for modifying a role.
Resource Association	Resource association for this role. The syntax must have a resource name, resource association description, and the entitlement value separated by semi colons(;). The format is *Resource Name in LDAP format;Resource association description;Entitlement Value*. For example, `cn=Group,cn=ResourceDefs,cn=RoleConfig,cn=AppConfig,cn=User Application Driver,cn=driverset1,o=system;Test Description;{ "ID":"25713f856ecfb24986ebc35bcd581906","ID2":"CN=Administrators,CN=Builtin,DC=idmseup2,DC=org"}`. For a static resource, the entitlement value is not required. Multiple resource-association elements can be added to associate multiple resources with this role.
Revoke Approver	Role assignment for this role. The syntax must have a role name, role assignment description, and the Role relationship separated by semi colons(;). For example, Role Name in LDAP format;Role assignment description;Relationship. Relationship can be one of child or parent. Multiple role-association elements can be added to assign multiple roles to this role.
Revoke Quorum	Minimum percentage of approval required for revoking a role.

Move Destination Object

Causes an object in the destination datastore to be moved.

Fields

Class Name

(Optional) Specify the class name of the object to move into the destination data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object to Move

Select the object to be moved. This object can be the current object, or can be specified by a DN or an association.

Container to Move to

Select the container to receive the object. This container is specified by a DN or an association.

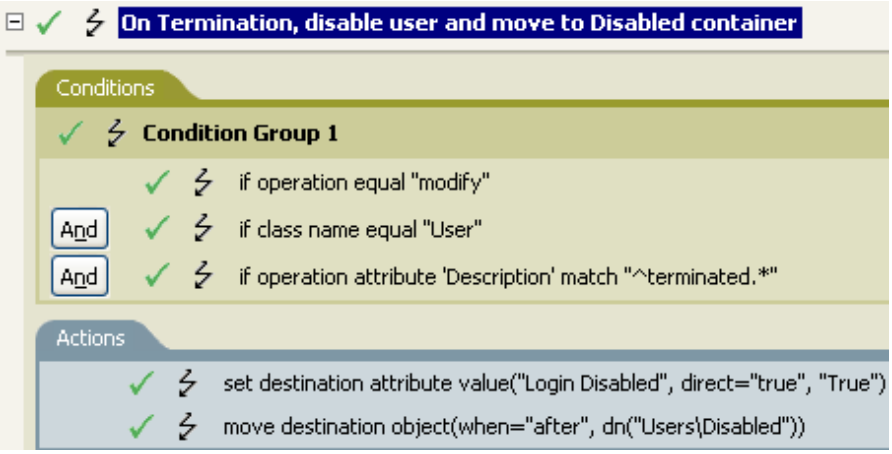
DN or Association

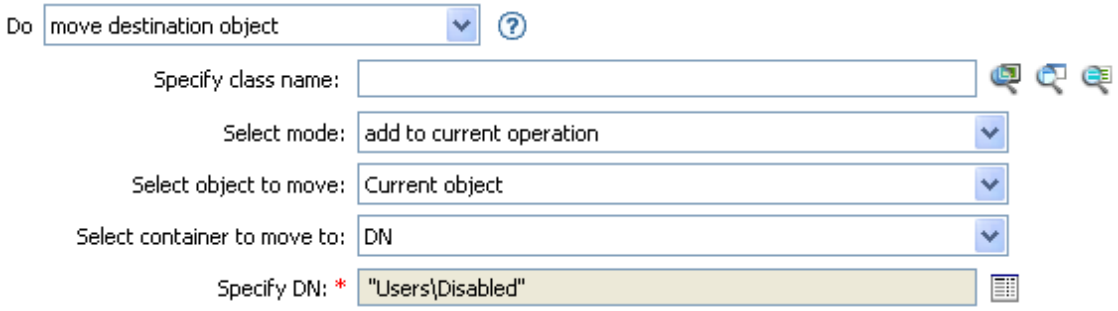
Specify whether the DN or association of the container is used.

IMPORTANT: When using this action in a policy on the Subscriber channel, do not use DN to specify the container to which you want to move the object. Instead, use association.

Example

The example contains a single rule that disables a user's account and moves it to a disabled container when the Description attribute indicates it is terminated. The policy is named Disable User Account and Move When Terminated, and it is available for download from the OpenText Support Web site. For more information, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view this policy in XML, see `005-Command-DisableMoveOnTermination` (../samples/005-Command-DisableMoveOnTermination.xml).





The policy checks to see if it is a modify event on a User object and if the attribute Description contains the value of terminated. If that is the case, then it sets the attribute of `Login Disabled` to `True` and moves the object into the `User\Disabled` container.

Move Source Object

Causes an object in the source datastore to be moved.

Fields

Class Name

(Optional) Specify the class name of the object to move into the source data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Object to Move

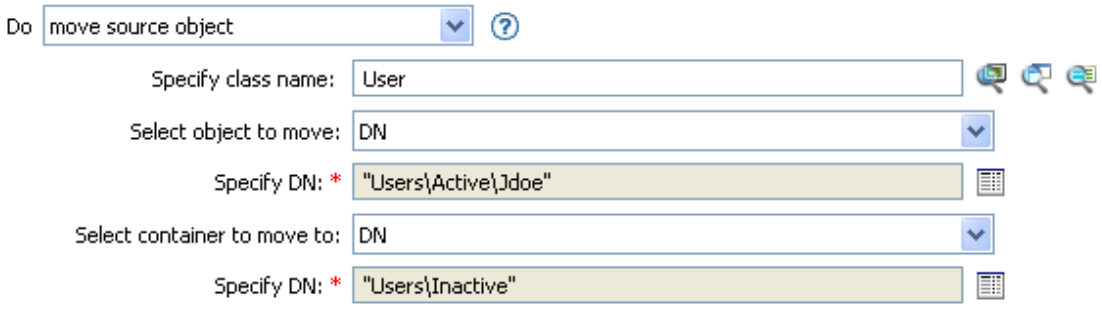
Select the object to be moved. This object can be the current object, or it can be specified by a DN or an association.

Select Container

Select the container to receive the object. This container is specified by a DN or an association.

IMPORTANT: When using this action in a policy on the Publisher channel, do not specify the DN of the container that will receive the object. Instead, use association.

Example



Reformat Operation Attribute

Reformats all values of an attribute within the current operation by using a pattern.

Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Value Type

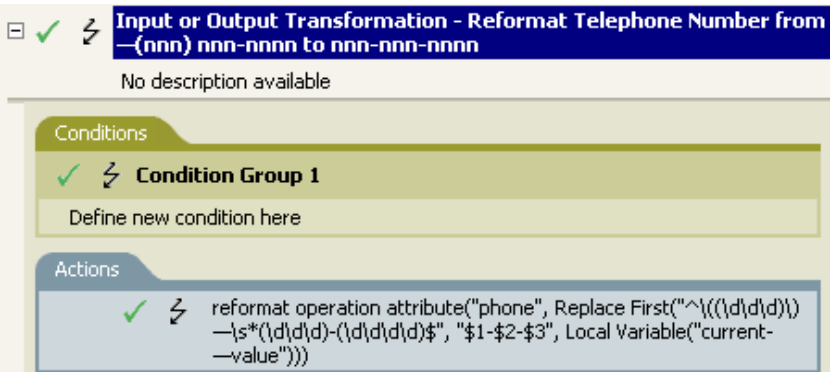
Specify the syntax of the new attribute value.

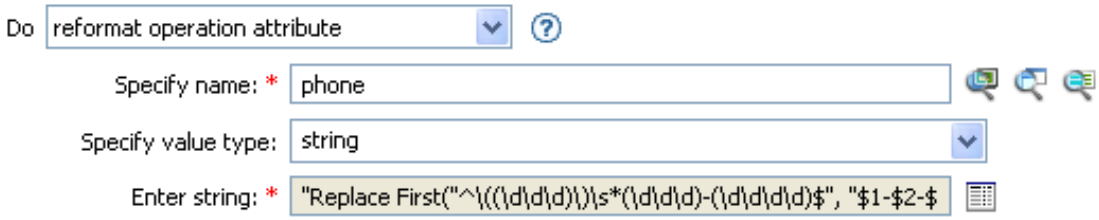
Value

Specify a value to use as a pattern for the new format of the attribute values. If the original value is needed to constructed the new value, it must be obtained by referencing the local variable current-value.

Example

The example reformats the telephone number. It changes it from (nnn)-nnn-nnnn to nnn-nnn-nnnn. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn" on page 122. To view the policy in XML, see `predef_transformation_reformat_telephone1.xml` (../samples/predef_transformation_reformat_telephone1.xml).





The action reformat operation attribute changes the format of the telephone number. The rule uses the Argument Builder and regular expressions to change how the information is displayed.

```
Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3")
    Local Variable("current-value")
```

Remove Association

Sends a remove association command to the Identity Vault.

Fields

Mode

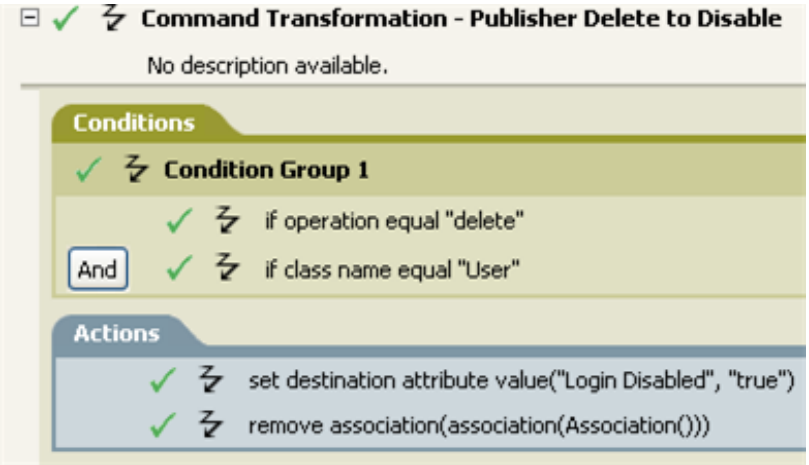
Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

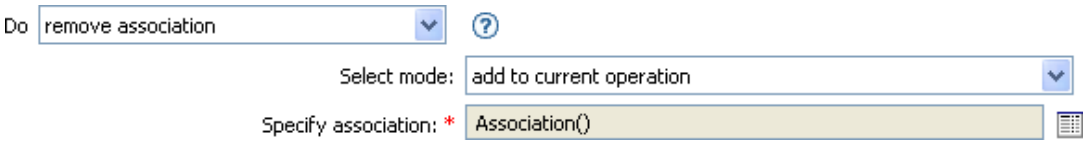
Association

Specify the value of the association to be removed.

Example

The example takes a Delete operation and disables the User object instead. The transforms an event. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Publisher Delete to Disable" on page 110. To view the policy in XML, see `predef_command_delete_to_disable.xml` (../samples/predef_command_delete_to_disable.xml).





When a Delete operation occurs for a User object, value of the Login Disabled attribute is set to True and the association is removed from the object. The association is removed because the associated object in the connected application no longer exists.

Remove Destination Attribute Value

Removes an attribute value from an object in the destination data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

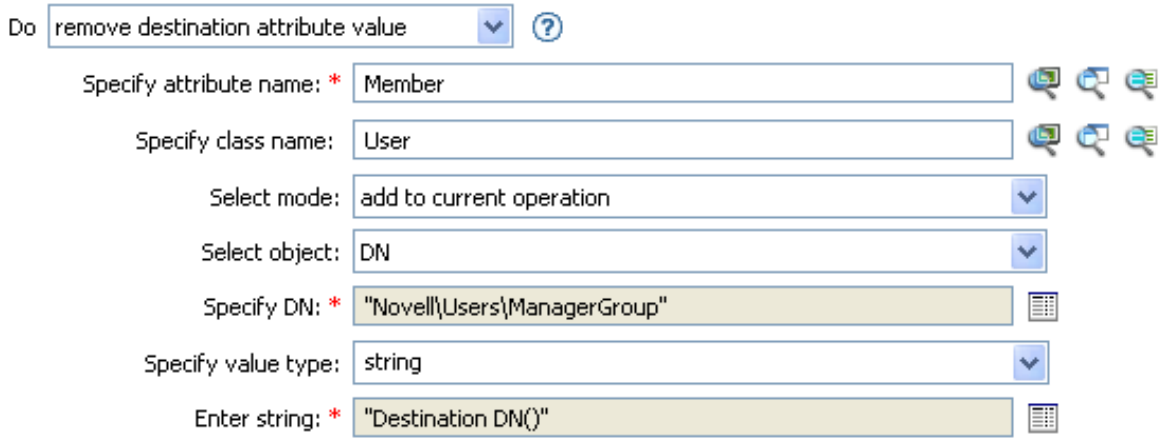
Value Type

Specify the attribute syntax of the value you want to remove.

Value

Specify the attribute value of the value you want to remove.

Example



Remove Named Password

Removes the named password for the DirXML-Driver or DirXML-DriverSet object using the value specified by `arg-dn`. The name of the Named Password name is specified by `<arg-string>`. This operation is applicable only for the current Driver and DriverSet objects. In addition to the Add Named Password token, these tokens allows a policy to add a password and then remove it after use.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Example

The Remove Named Password action supports the following string arguments:

String Name	Description
Name	Name of the named password.

Remove Role

Initiates a request to the Roles Based Provisioning Module (RBPM) to revoke the specified role (in the Role DN field) from the specified user (in the Authorized User DN field). This field is only available if the OpenText Identity Manager server version is set to 3.6 or later. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-remove-role`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Role DN

Specify the name of the role to revoke, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the role assignment, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN or Association

Select the DN or association as the target object.

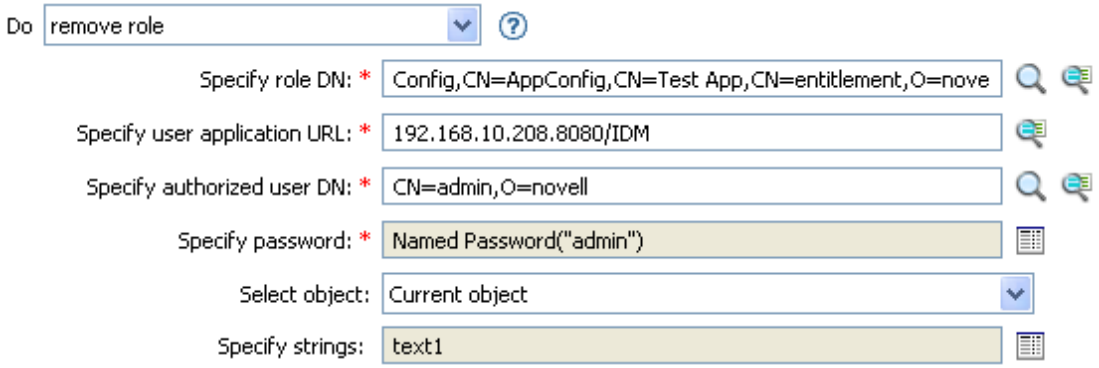
Strings

(Optional) Specify additional argument strings for the Role assignment request. You can enter the strings manually, or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Remove Role action supports the following string arguments:

String Name	Description
role-assignment-type	The type of the role assignment. You can choose from the following options: ◆ USER_TO_ROLE ◆ GROUP_TO_ROLE ◆ CONTAINER_TO_ROLE ◆ ROLE_TO_ROLE Default: USER_TO_ROLE.
description	A description of the reason for the request used for auditing and (if necessary) approval purposes. Default: Request generated by the policy.
effective-time	The time (in CTIME format) the role assignment should become effective. Default: now
CorrelationID	An identifier to correlate the role revocation process. Default: Operation event Correlation ID If no value is specified for the argument, it uses the default value. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.

Example



Remove Resource

Initiates a request to the Roles Based Provisioning Module (RBPM) to revoke the specified resource (in the Resource DN field) from the specified user (in the Authorized User DN field). This field is only available if the OpenText Identity Manager server version is set to 4.0.2 or later. You can specify optional arguments to the resource assignment request by using the <arg-string> argument. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-remove-resource`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Resource DN

Specify the name of the resource to revoke, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server hosting the Roles Based Provisioning module. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Instance GUID

Specify the ID of the resource assignment for users for revoking a single instance of a multivalued resource. If you do not specify any value, all the instances of the resource are revoked. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the name of the user authorized to request the resource assignment, in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is 0.

Password

Specify the authorized user password. You can enter a clear text password (not recommended) or use the Argument Builder to specify a Named Password.

Object

Select the target object type. This object can be the current object, or can be specified by a DN or an association.

DN or Association

Select the DN or association as the target object.

Strings

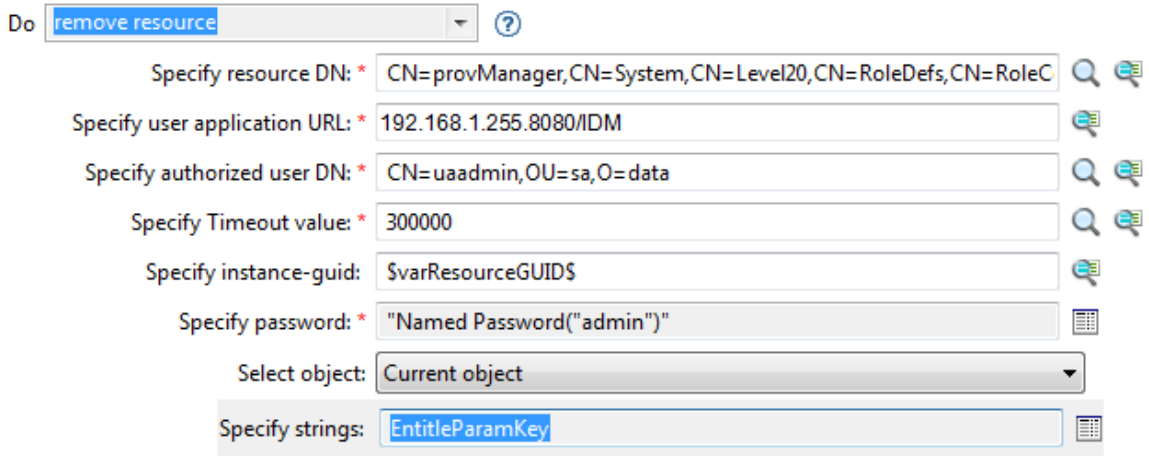
(Optional) Specify additional argument strings for the Resource assignment request. You can enter the strings manually, or select the **Edit the Strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Remove Resource action supports the following string arguments:

String Name	Description
description	A description of the reason for the request used for auditing and (if necessary) approval purposes. Default: Request generated by policy.
CorrelationID	An identifier to correlate the resource revocation process. Default: Operation event Correlation ID If no value is specified for the argument, it uses the default value. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.
EntitleParamKey	Specifies the value of the entitlement which needs to be removed in JSON format.

Example

In the following example, for the "**remove resource**" action, provide values for all parameters including Instance guid. Create a parameter called `EntitleParamKey` in the **Specify Strings** field. The value of this field is the name of the entitlement which is to be removed in JSON format.



Remove Source Attribute Value

Removes the specified value from the named attribute on an object in the source data store.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

DN or Association

Select the DN or association as the target object.

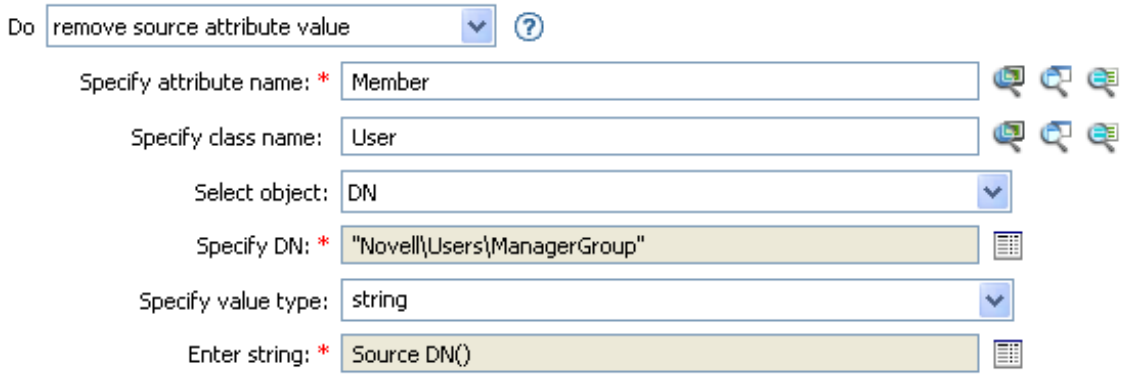
Value Type

Specify the syntax of the attribute value to be removed.

String

Specify the attribute value to be removed.

Example



Rename Destination Object

Renames an object in the destination data store.

Fields

Class Name

(Optional) Specify the class name of the object to rename in the destination data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

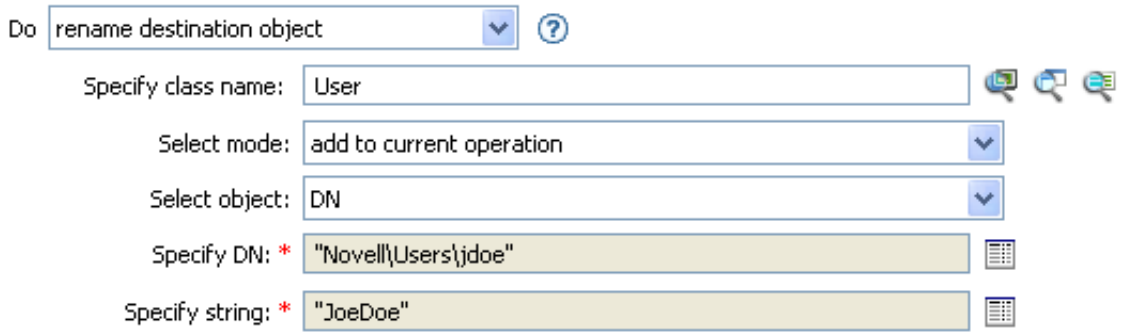
DN or Association

Select the DN or association as the target object.

String

Specify the new name of the object.

Example



Rename Operation Attribute

Renames all occurrences of an attribute within the current operation.

Fields

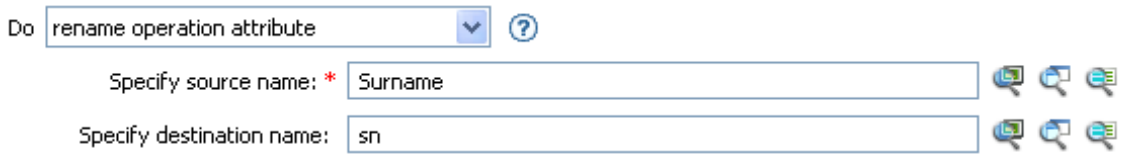
Source Name

Specify the original attribute name. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Destination Name

Specify the new attribute name. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example



Rename Source Object

Renames an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to rename in the source data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Select Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

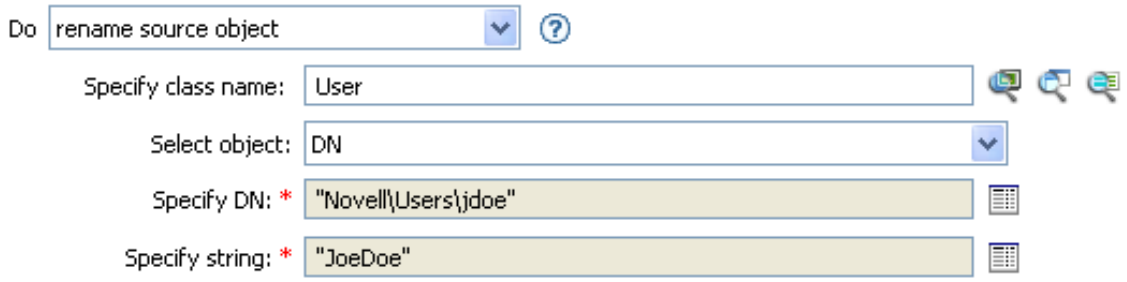
DN or Association

Select the DN or association as the target object.

String

Specify the new name of the object.

Example



Send Email

Sends an e-mail notification. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-send-email`. For more information about local error variables, see "Local Variable Selector" on page 67.

Considerations

Consider the following points when creating a policy with `<do-send-email>` action:

- Identity Manager 4.8 and onwards, the engine checks to see if the mail server supports SSL, and if it supports, then it utilizes SSL. Therefore the mail server certificate must be added to the OpenText Identity Manager engine keystore.
- Prior to Identity Manager 4.9, the application only supported basic authentication, which required users to provide SMTP server account credentials (user ID and password) to send emails through the mail server. With Identity Manager 4.9 and later versions, users can authenticate to the mail server using either basic or modern authentication methods.
- Azure Active Directory uses modern authentication provided by the Office 365 OAuth authentication service. So if you are using Microsoft Outlook on Azure Active Directory as your mail server setup, then you need to register your application with Azure AD first. For more information on how to register your application, see Register an application with the Microsoft identity platform (https://learn.microsoft.com/en-us/entra/identity-platform/quickstart-register-app).
- When creating a policy with the `<do-send-email>` action, you must set up a rule to use **named passwords** in the driver policy. This is because the engine uses the named passwords to read the SMTP server account password and the client secret in order to send email notifications. Each password is accessed via a key, for example, the `direxml.engine.smtp` key stores the SMTP server account password for basic authentication, while the `dirxml.engine.smtp.clientsecret` key stores the client secret for modern authentication. Both of these keys are stored in the named passwords at the active driverset level.

Fields

ID

(Optional) Specify the User ID in the SMTP system sending the message. Supports variable expansion. For more information, see "Variable Selector" on page 31.

NOTE: The **ID** field is required for modern authentication. You must specify the email account address that will be used as the originating e-mail address, which is the same as address defined in the "from" string name.

Server

Specify the SMTP server address. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Message Type

Select the e-mail message type.

Password

(Optional) Specify the SMTP server account password.

IMPORTANT: You can store the SMTP server account password as a Named Password on the driver object. This allows the password to be encrypted; otherwise, you enter the password and it is stored in clear text. For more information on Named Passwords, see "Securely Storing Driver Passwords with Named Passwords" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

Strings

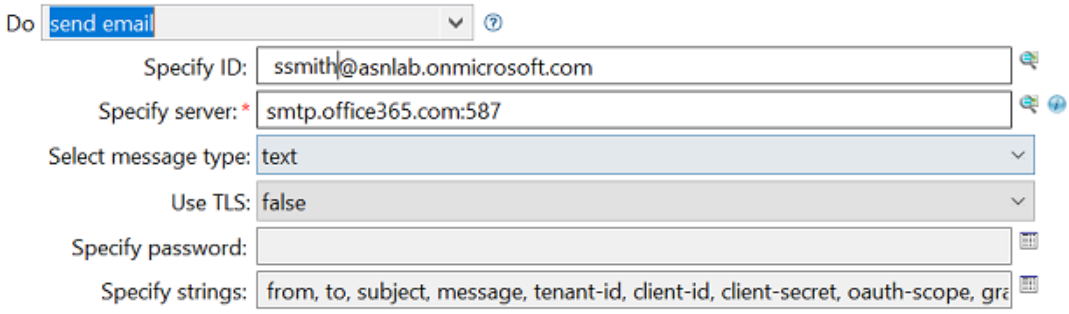
Specify the values containing the various e-mail addresses, subject, and message. You can enter the strings manually, or select the **Edit the strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The Send Email action supports the following string arguments:

String Name	Description
The following string arguments are applicable if the mail server is set for either basic or modern authentication:	
bcc	Adds the address to the list of BCC e-mail recipients; multiple instances are allowed. Can contain a comma-separated list of recipients.
cc	Adds the address to the list of CC e-mail recipients; multiple instances are allowed. Can contain a comma-separated list of recipients.
custom-smtp-header	Specifies a custom SMTP header to add to the e-mail message.
encoding	Specifies the character encoding to use for the e-mail message.
from	Specifies the address to be used as the originating e-mail address.
message	Specifies the content of the e-mail message.
reply-to	Specifies the address to be used as the e-mail message reply address.
subject	Specifies the e-mail subject.
to	Adds the address to the list of e-mail recipients; multiple instances are allowed. Can contain a comma-separated list of recipients.
The following string arguments are only applicable if the mail server is set for modern authentication:	
tenant-id	Specifies the Global Unique Identifier (GUID) for your registered account on Azure Active Directory. Also referred to as the Tenant (Directory) ID. Use the Azure portal to obtain the value.
client-id	Specifies the unique ID assigned by Azure Active Directory to your application. Also referred to as the Application (client) ID. Use the Azure portal to obtain the value.

String Name	Description
client-secret	Specifies a secret string that the application uses to prove its identity when requesting an access token. This encrypted value is unique for your application. Use the Azure portal to obtain the value.
oauth-scope	Specifies the resource for which the token is granted during token redemption. It defines what the application can access on behalf of the user. It should be `https://outlook.office365.com/.default` for SMTP. The access token issued to the application is limited to the scope granted.
grant-type	Used by the application to obtain an access token to access its own resources, not on behalf of a user. It is set to `client_credentials` by default and should not be modified.
requestURL	Specifies the request URI of the SMTP server, to which the application sends authentication requests containing client secret, scope, grant type, and auth mechanism in the payload. For example, `https://login.microsoftonline.com/{tenantID}/oauth2/v2.0/token`
auth-mechanism	Specifies the format to use to encode and transmit the access token to the SMTP server. The access token authenticates the user's Outlook account. It is set to XOAUTH2 by default and should not be modified.

Example



Send Email from Template

Generates an e-mail notification by using a SMTP notification configuration object, e-mail template object, and replacement tokens. It reads the target SMTP server along with the credentials for authentication (except for the password, which when needed is specified by <arg-password>) and the originating address from the SMTP notification configuration object. If e-mail sender's address is specified, the action overrides the originator's address specified in the e-mail server configuration. The subject and e-mail message are created using the template object and template replacement tokens. Replacement tokens are declared within an `<arg-string>` element and tag name attribute. The value of `<arg-string>'s` tag attribute is interpreted as `html`, if the attribute is enclosed within `<use-html>...</use-html>` tags. Reserved replacement tokens specify the various recipient addresses. If a policy containing this action encounters an error, Designer generates this error as a local variable: `error.do-send-email-from-template`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

Notification DN

Specify the slash form DN of the SMTP notification configuration object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Template DN

Specify the slash form DN of the e-mail template object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Password

(Optional) Specify the SMTP server account password. Select the **Edit the arguments** icon to open the Argument Builder and specify the password argument.

IMPORTANT:

- You can store the SMTP server account password as a Named Password on the driver object. This allows the password to be encrypted; otherwise, you enter the password and it is stored in clear text. For more information on Named Passwords, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

- If you do not specify a password, one of the following conditions will apply depending on the authentication method used by the mail server:
  - For basic authentication, the engine reads the server password from **named passwords** using the `direxml.engine.smtp` key, which is stored at the active driverset level.
  - For modern authentication, the client secret is used instead of server password. In this case, the engine reads the client secret from **named passwords** using the `dirxml.engine.smtp.clientsecret` key, which is also stored in the active driverset level.

Strings

Specify additional string arguments for the email message. You can enter the strings manually, or select the **Edit the strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

Send Email from Template supports the following string arguments that you can use to specify the various email addresses.

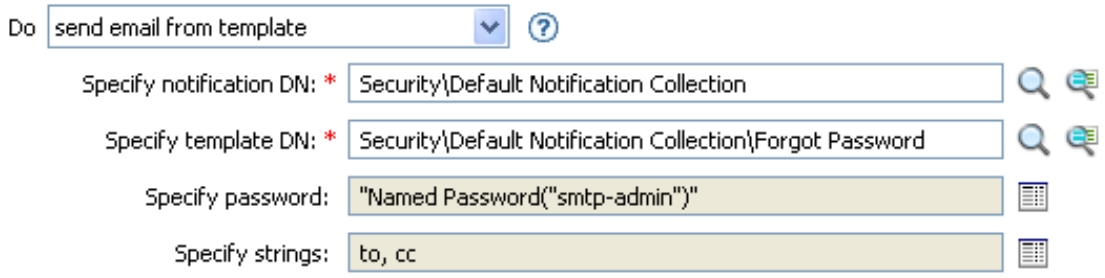
String Name	Description
from	Specifies the address of the sender of the e-mail. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.
to	Adds the address to the list of e-mail recipients. Multiple instances are allowed. Can contain a comma-separated list of recipients.
cc	Adds the address to the list of CC e-mail recipients; multiple instances are allowed. Can contain a comma-separated list of recipients.
bcc	Adds the address to the list of BCC e-mail recipients; multiple instances are allowed. Can contain a comma-separated list of recipients.
attachment	Adds an attachment to the e-mail. Multiple attachments to a single e-mail are allowed. **NOTE:** This string argument is not available in the Policy Builder user interface of this version.
reply-to	Specifies the address to be used as the e-mail message reply address.
encoding	Specifies the character encoding to use for the e-mail message.
custom-smtp-header	Specifies a custom SMTP header to add to the e-mail message.

In addition to the reserved field names listed above, Send Email from Template supports Global Configuration Values (GCVs) for creating the desired string.

NOTE: To include HTML in a string argument, you can enclose the HTML in the `<use-html></use-html>` tags within the string. The HTML enclosed by the tags must comply with the XHTML standard.

Each template can also define fields that can be replaced in the subject and body of the e-mail message.

Example


Do
send email from template
Specify notification DN: *
Security\Default Notification Collection
Specify template DN: *
Security\Default Notification Collection\Forgot Password
Specify password:
"Named Password("smtp-admin")"
Specify strings:
to, cc

Set Default Attribute Value

Adds default values to the current operation (and optionally to the current object in the source data store) if no values for that attribute already exist. It is only valid when the current operation is Add.

Fields

Attribute Name

Specify the name of the default attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Write Back

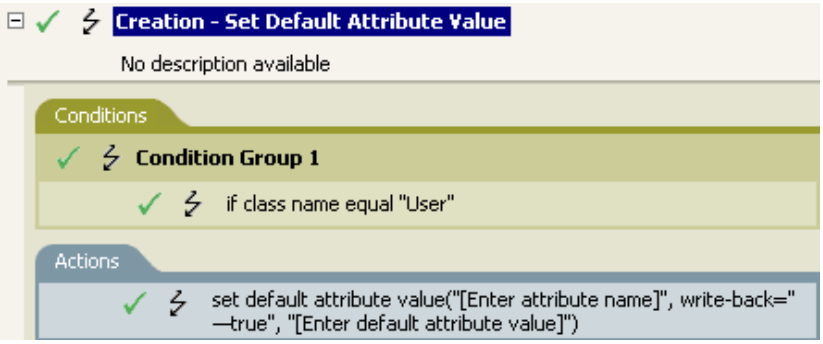
Select whether or not to also write back the default values to the source data store.

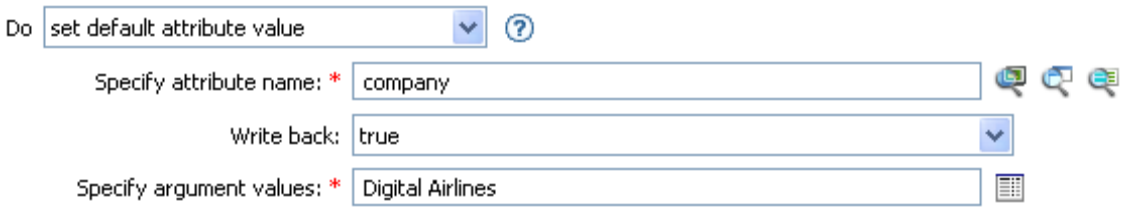
Argument Values

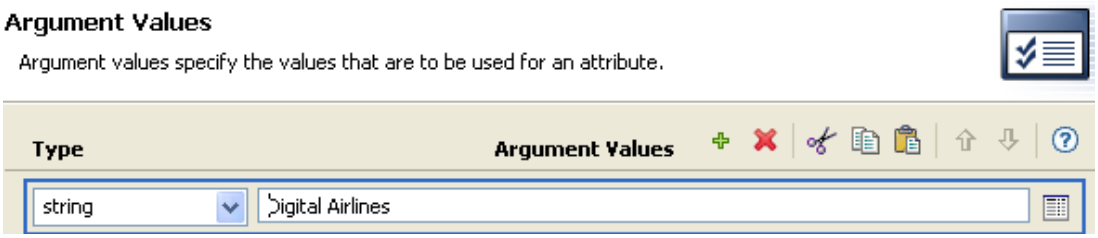
Specify the default values of the attribute.

Example

The example sets the default value for the company attribute. You can set the value for an attribute of your choice. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Creation - Set Default Attribute Value" on page 115. To view the policy in XML, see `predef_creation_set_default_attribute_value.xml` (../samples/predef_creation_set_default_attribute_value.xml).







To build the value, the Argument Value List Builder is launched. See "Argument Value List Builder" on page 54 for more information on the builder. You can set the value to what is needed. In this case, we used the Argument Builder and set the text to be the name of the company.

Set Destination Attribute Value

Adds a value to an attribute on an object in the destination data store, and removes all other values for that attribute.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object in the destination data store. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Value Type

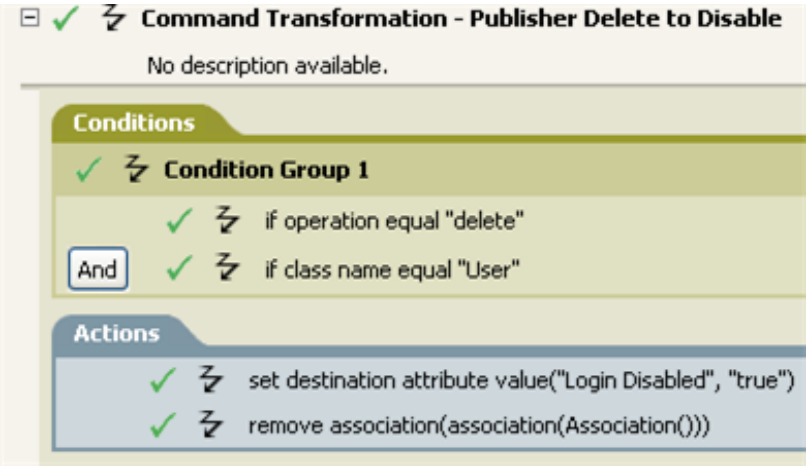
Select the syntax of the attribute value to set.

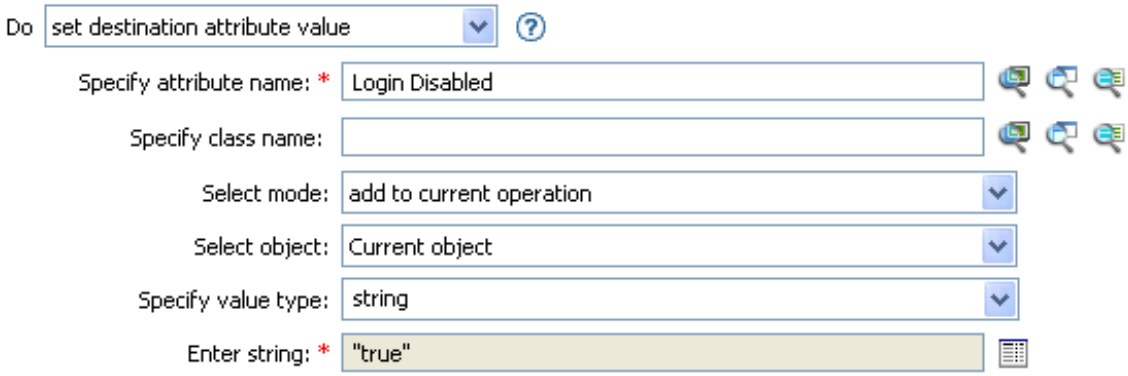
String

Specify the attribute values to set.

Example

The example takes a Delete operation and disables the User object instead. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Publisher Delete to Disable" on page 110. To view the policy in XML, see `predef_command_delete_to_disable.xml` (../samples/predef_command_delete_to_disable.xml).





The rule sets the value for the attribute of Login Disabled to true. The rule uses the Argument Builder to add the text of true as the value of the attribute. See "Argument Builder" on page 43 for more information about the builder.

Set Destination Password

Sets the password for an object in the destination data store.

Fields

Class Name

(Optional) Specify the class name for the object to set the password on in the destination data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

New Password

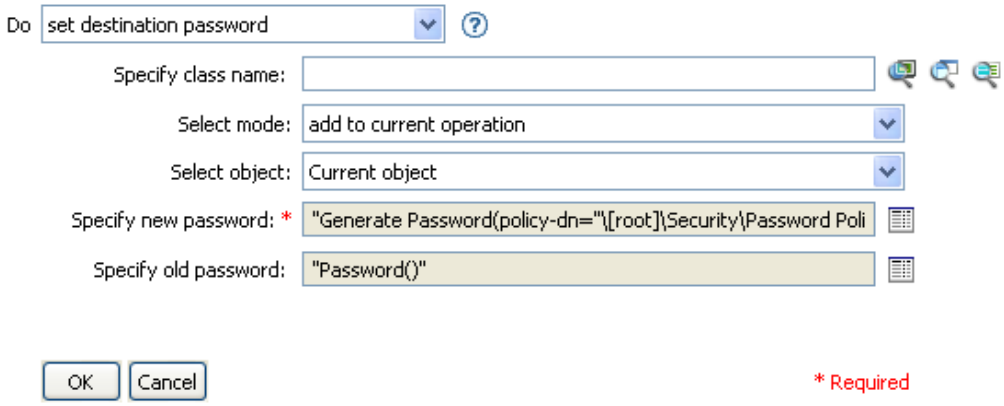
Specify the password to be set.

Old Password

Specifies the old password, which is used to confirm that you have rights to change the password.

NOTE: This field is only available if the OpenText Identity Manager server version is set to 3.6 or later.

Example



Set Local Variable

Sets a local variable.

Fields

Variable Name

Specify the name of the new local variable. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Scope

Select the scope of the local variable. This can be set to the driver or to the policy. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Variable Type

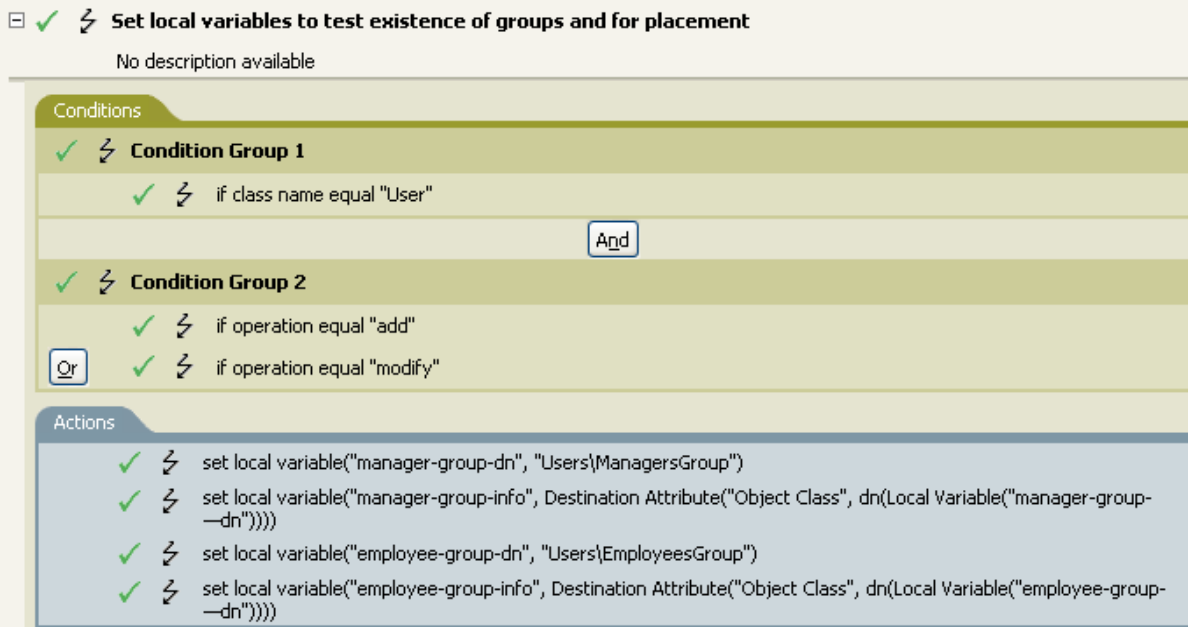
Select the type of local variable. This can be a string, an XPath 1.0 node set, or a Java* object.

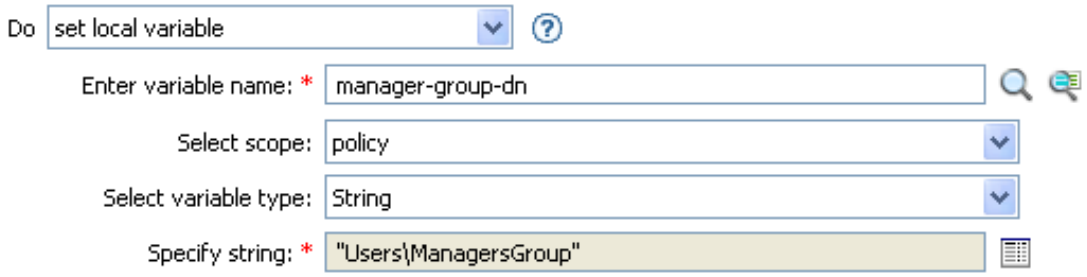
String

Specify the attribute values to set.

Example

The example adds a User object to the appropriate Employee or Manager group based on Title. It also creates the group, if needed, and sets up security equal to that group. The policy name is Govern Groups for User Based on Title, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).





The local variable is set to the value that is in the User object's destination attribute of Object Class plus the Local Variable of manager-group-info. The Argument Builder is used to construct the local variable. See "Argument Builder" on page 43 for more information.

Set Named Password

Sets a named password for a DirXML-Driver or DirXML-DriverSet object using the value specified by `arg-password`. The target object is specified by `arg-dn`. The name and display name of the named password are specified by `<arg-string>'s`.

Fields

Disabled

Specify whether the action is enabled or disabled, where true indicates enabled and false indicates disabled. The default value is false.

No Trace

Specify whether this action should be traced during the execution of the policy. The default value is false, which disables the trace.

Example

The Set Named Password action supports the following string arguments:

String Name	Description
Name	Name of the named password.
Display Name	Display name of the named password. Default: name of the named password

Set Operation Association

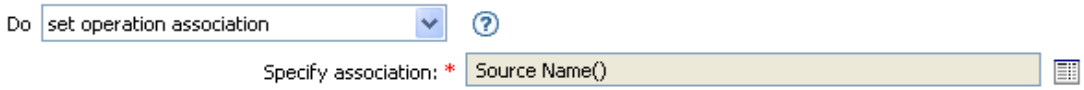
Sets the association value for the current operation.

Fields

Association

Provide the new association value.

Example



Set Operation Class Name

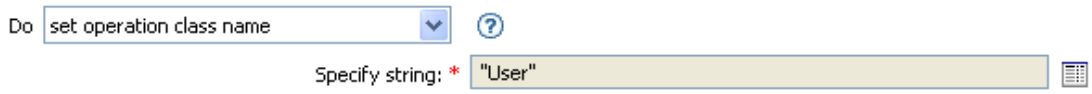
Sets the object class name for the current operation.

Fields

String

Specify the new class name.

Example



Set Operation Destination DN

Sets the destination DN for the current operation.

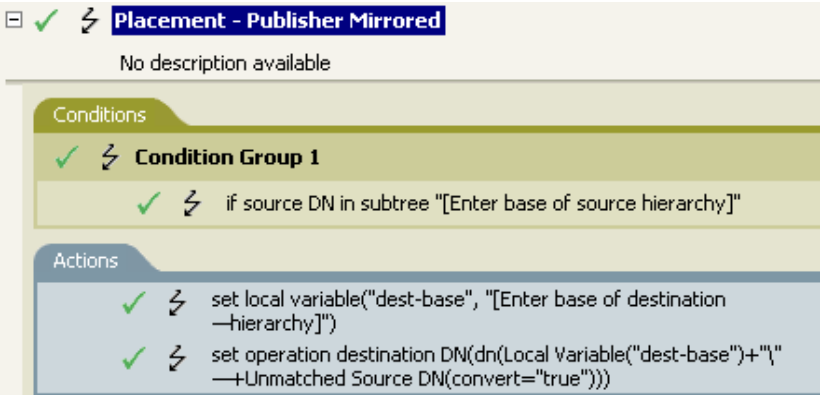
Fields

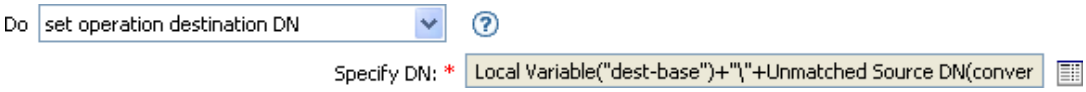
DN

Specify the new destination DN.

Example

This example places the objects in the Identity Vault, by using the structure that is mirrored from the connected system. You need to define at what point the mirroring begins in the source and destination data stores. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Placement - Publisher Mirrored" on page 131. To view the policy in XML, see `predef_place_pub_mirrored.xml` (../samples/predef_place_pub_mirrored.xml).





The rule sets the operation destination DN to be the local variable of the destination base location plus the source DN.

Set Operation Property

Sets an operation property. An operation property is a named value that is stored within an operation. It is typically used to supply additional context that might be needed by the policy that handles the results of an operation.

Fields

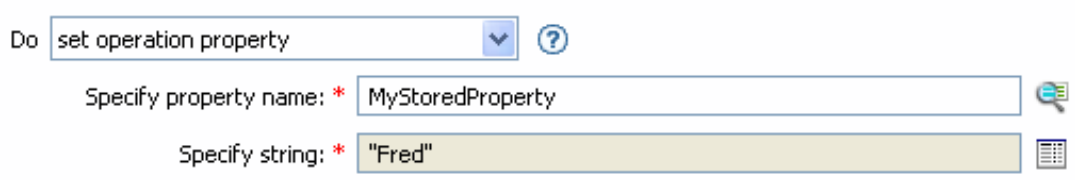
Property Name

Specify the name of the operation property. Supports variable expansion. For more information, see "Variable Selector" on page 31.

String

Specify the name of the string.

Example



Set Operation Source DN

Sets the source DN for the current operation.

Fields

DN

Specify the new source DN.

Example

Set Operation Template DN

Sets the template DN for the current operation to the specified value. This action is only valid when the current operation is Add.

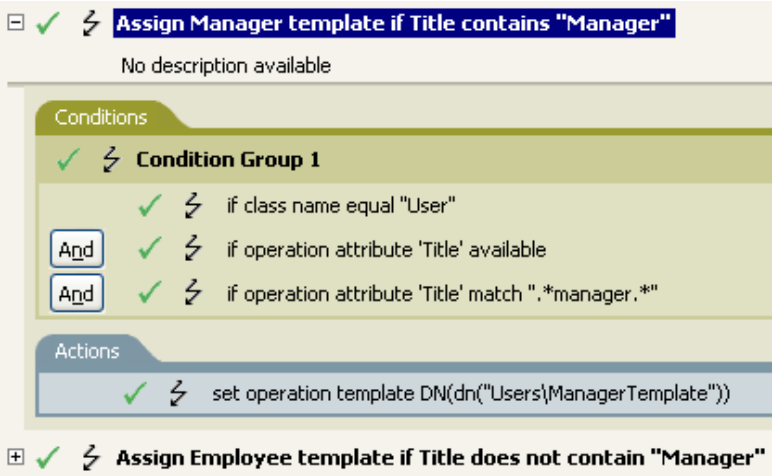
Fields

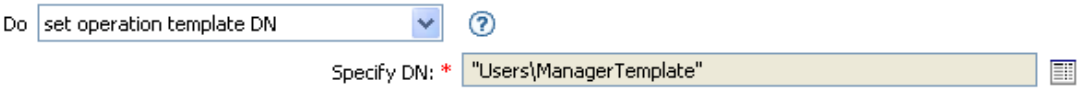
DN

Specify the template DN.

Example

The example applies the Manager template if the Title attribute contains the word Manager. The name of the policy is Policy: Assign Template to User Based on Title, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Create-AssignTemplateByTitle.xml` (../samples/003-Create-AssignTemplateByTitle.xml).





The template Manager Template is applied to any User object the has the attribute of Title available and contains the word Manager somewhere in the title. The policy uses regular expressions to find all possible matches.

Set Source Attribute Value

Adds a value to an attribute on an object in the source data store, and removes all other values for that attribute.

Fields

Attribute Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object in the source data store. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Value Type

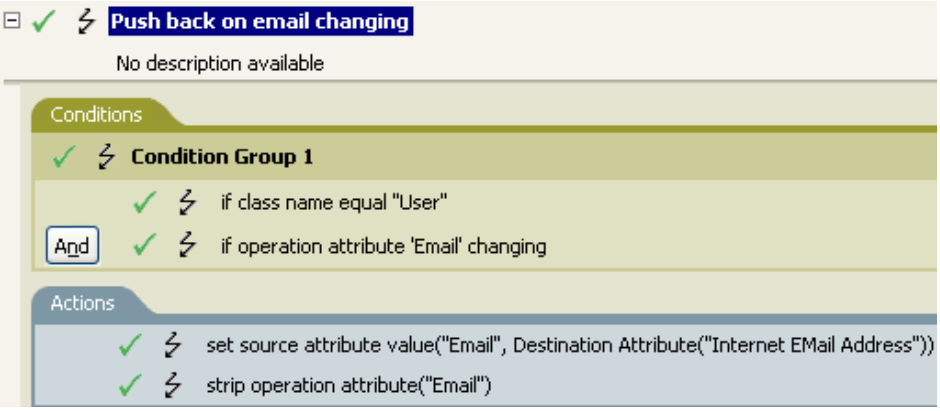
Select the syntax of the attribute value.

Value

Specify the attribute value to be set.

Example

The example detects when an e-mail address is changed and sets it back to what it was. The policy name is Policy: Reset Value of the E-mail Attribute, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Input_PushBackOnEmail` (../samples/001-Input-PushBackOnEmail.xml).



Do set source attribute value

Specify attribute name: * Email

Specify class name:

Select object: Current object

Specify value type: string

Enter string: * Destination Attribute("Internet EMail Address")

The action takes the value of the destination attribute Internet EMail Address and sets the source attribute of Email to this same value.

Set Source Password

Sets the password for an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to set the password on in the source data store. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Object

Select the target object. This object can be the current object, or can be specified by an DN or an association.

New Password

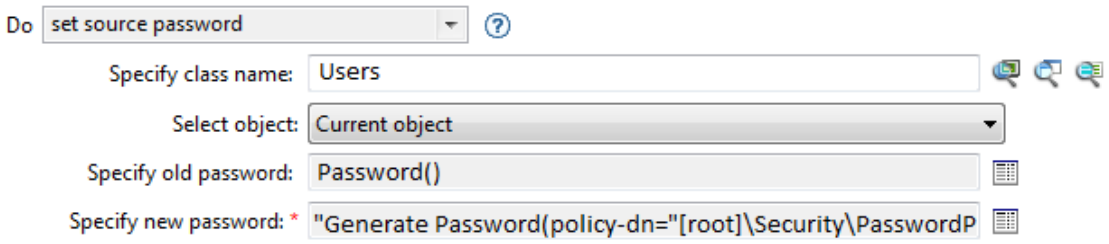
Specify the password to be set.

Old Password

Specifies the old password, which is used to confirm that you have rights to change the password.

NOTE: This field is only available if the OpenText Identity Manager server version is set to 3.6 or later.

Example



Set SSO Credential

Sets the SSO credential when a user object is created or when a password is modified. This action is part of the Credential Provisioning policies. For more information, see the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-set-sso-credential`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Target User DN

Specify the DN of the target users.

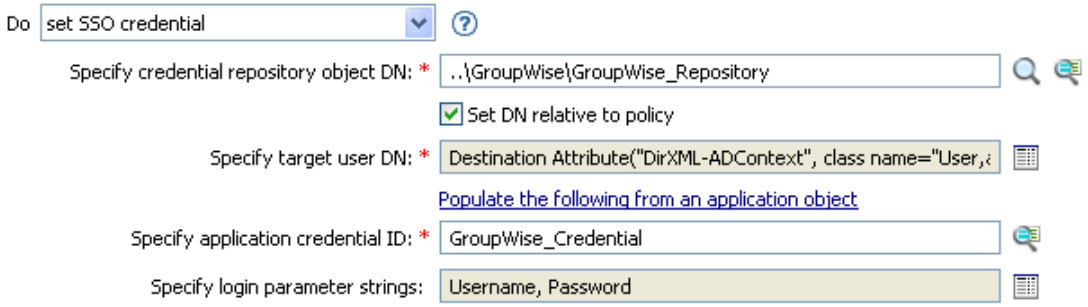
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object. You can enter the strings manually, or select the **Edit the strings** icon to open the String Builder and specify the strings. For more information about the Named String Builder, see "String Builder" on page 58.

Example



Set SSO Passphrase

Sets the SecureLogin passphrase and answer when a User object is provisioned. This action is part of the Credential Provisioning policies. For more information, see the *OpenText™ Identity Manager CE 24.4 (v4.10) Credential Provisioning Guide*. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-set-sso-passphrase`. For more information about local error variables, see "Local Variable Selector" on page 67.

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Target User DN

Specify the DN of the target users.

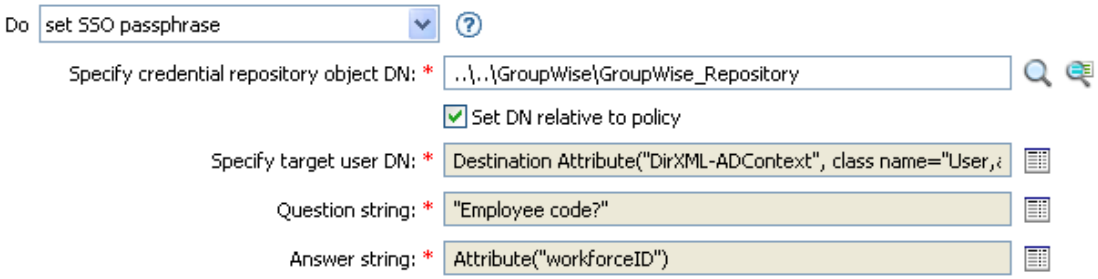
Question String

Specify the SecureLogin passphrase question.

Answer String

Specify the SecureLogin passphrase answer.

Example



The SecureLogin passphrase question and answer are stored as strings in the policy.

Set XML Attribute

Sets an XML attribute on a set of elements selected by an XPath expression.

Fields

Name

Specify the name of the XML attribute. This name can contain a namespace prefix if the prefix has been previously defined in this policy. Supports variable expansion. For more information, see "Variable Selector" on page 31.

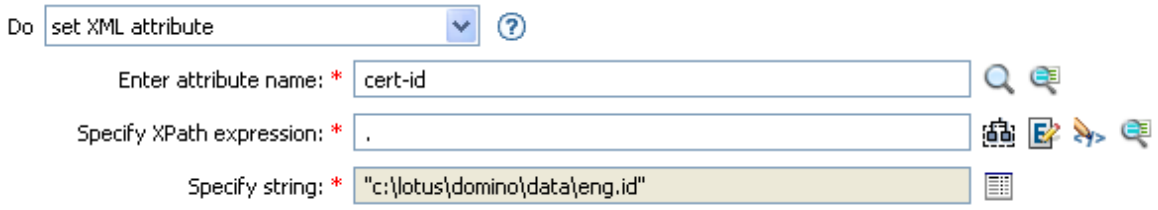
XPath Expression

XPath 1.0 expression that returns a node set containing the elements on which the XML attribute should be set. Supports variable expansion. For more information on variable expansion and XPath, see "XPath Expressions" on page 35.

String

Specify the value of the XML attribute.

Example



Start Workflow

Starts the workflow specified by workflow-id for the recipient DN on the User Application server specified by a URL and by using credentials specified by the ID and password. The recipient must be an LDAP format DN of an object in the directory served by the User Application server. The additional arguments to the workflow can be specified by named strings. Multiple values can be specified delimited by a semi-colon(;). If the semi-colon is part of the value, then it should be escaped by using a backslash(\). The number of the strings and the names used are dependent on the workflow to be started. If a policy containing this action encounters an error, Designer generates the error as the local variable `error.do-start-workflow`. For more information about local error variables, see "Local Variable Selector" on page 67.

Remark

There are some names that have special meaning and are available regardless of the workflow being started.

- **InitiatorOverrideDN:** The LDAP format DN of the initiator of the workflow, if other than the User used to authenticate.
- **CorrelationID:** An identifier used to correlate related workflows. By default, Operation event correlation is used. If no value is specified for the identifier, default value is used.

NOTE: `CorrelationID` is not available in the Policy Builder user interface of this version.

If any error occurs while starting the workflow, the error string is available to the enclosing policy in the local variable named `error.do-start-workflow`. Otherwise, that local variable is unavailable.

Fields

Provisioning Request DN

Specify the DN of the workflow to start in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

User Application URL

Specify the URL of the User Application server where the workflow will run. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Authorized User DN

Specify the DN of a user authorized to start workflows on the User Application server in LDAP format. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Timeout Value

Specify the number of milliseconds you want OpenText Identity Manager to try to establish a connection to the User Application server before timing out. The default value is `0`.

Authorized User Password

Specify the password of the authorized user to start workflows on the User Application server. Store the password as a Named Password on the driver object. This allows the password to be encrypted when it is stored.

Recipient DN

Specify the DN of the recipient of the workflow in LDAP format.

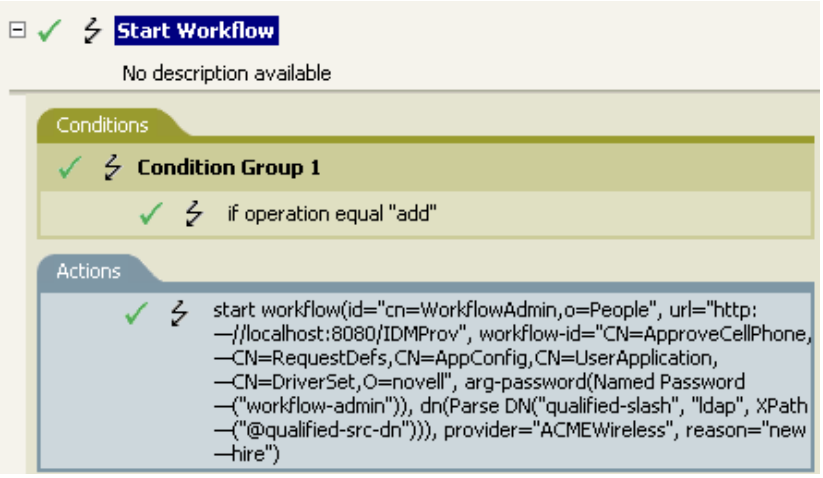
Strings

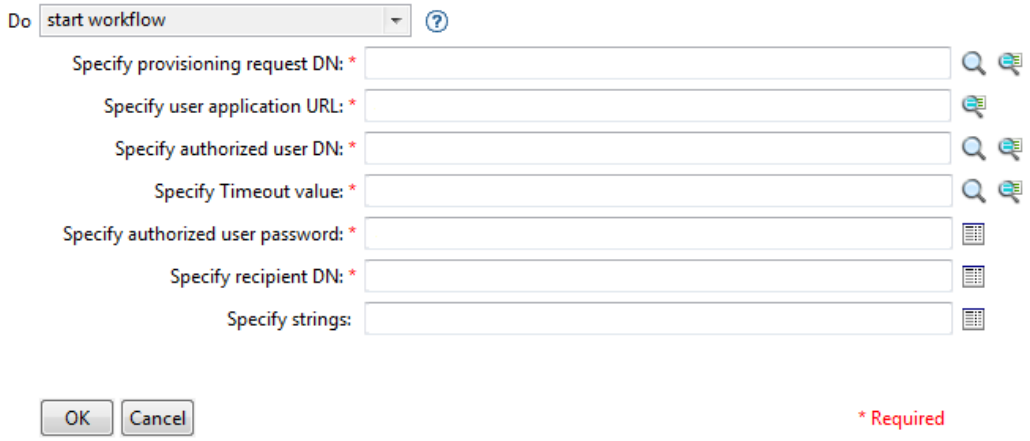
Specify the arguments for the workflow.You can enter the strings manually, or select the **Edit the strings** icon to open the Named String Builder and specify the strings. For more information about the Named String Builder, see "Named String Builder" on page 55.

The arguments are defined on the workflow. Depending on how the workflow is defined, some of the arguments might be required for the workflow to start.

Example

The following example starts a workflow process each time there in an Add operation. The workflow is a request for a cell phone. To view the policy in XML, see `start_workflow.xml` (../samples/start_workflow.xml).





Status

Generates a status notification.

Fields

Level

Specify the status level of the notification. The levels are error, fatal, retry, success, and warning. Supports variable expansion. For more information, see "Variable Selector" on page 31.

String

Provide the status message by using the Argument Builder.

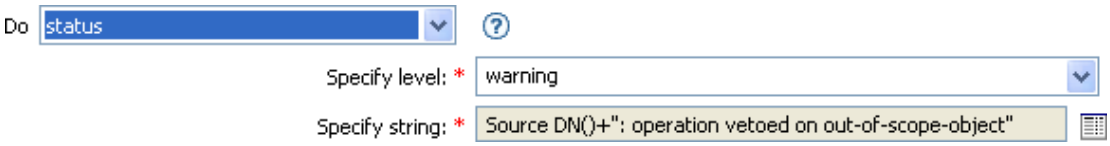
Remarks

If level is retry, then the policy immediately stops processing the input document and schedules a retry of the event currently being processed.

If the level is fatal, the policy immediately stops processing the input document and initiates a shutdown of the driver.

If a the current operation has an event-id, that event-id is used for the status notification; otherwise, there is no event-id reported.

Example



Strip Operation Attribute

Strips all occurrences of an attribute from the current operation.

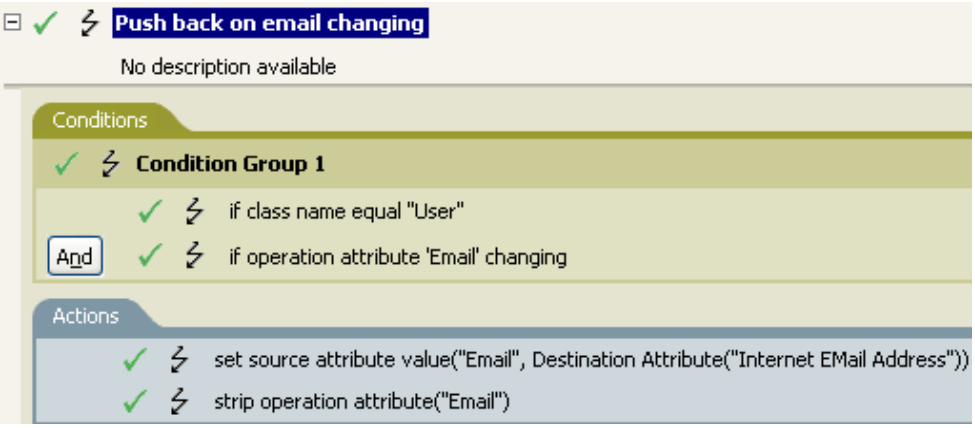
Fields

Name

Specify the name of the attribute to be stripped. Supports variable expansion. For more information, see “Variable Selector” on page 31.

Example

The example detects when an e-mail address is changed and sets it back to what it was. The policy name is Policy: Reset Value of the E-mail Attribute, and it is available for download from the OpenText Support Web site. For more information, see “Downloading Identity Manager Policies” in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Input-PushBackOnEmail.xml` (../samples/001-Input-PushBackOnEmail.xml).



The action strips the attribute of Email and keeps the value that was in the destination Email attribute.

Strip XPath Expression

Strips nodes selected by an XPath 1.0 expression.

Fields

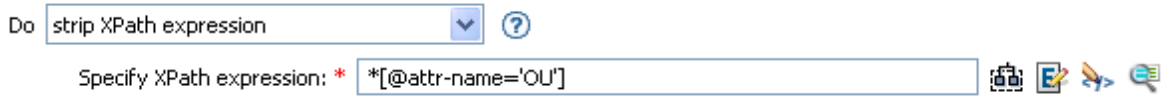
XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the nodes to be stripped. Supports variable expansion. For more information on variable expansion and XPath, see "Variable Selector" on page 31.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example



Trace Message

Sends a message to DSTRACE.

Fields

Level

Specify the trace level of the message. The default level is 0. The message only appears if the specified trace level is less than or equal to the trace level configured in the driver.

For information on how to set the trace level on the driver, see "Viewing OpenText Identity Manager Processes" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

Color

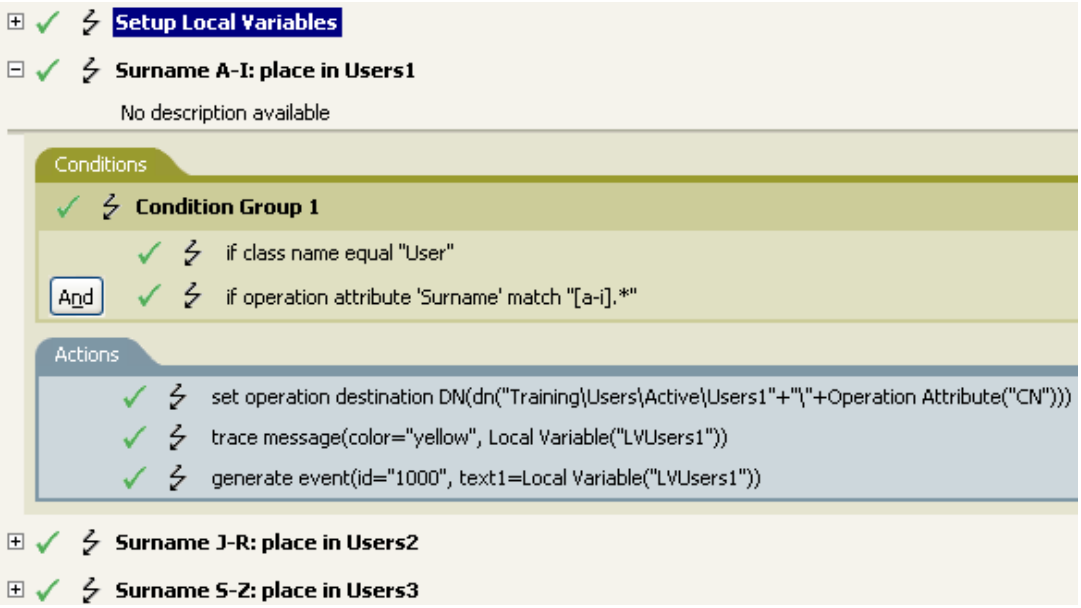
Select the color of the trace message.

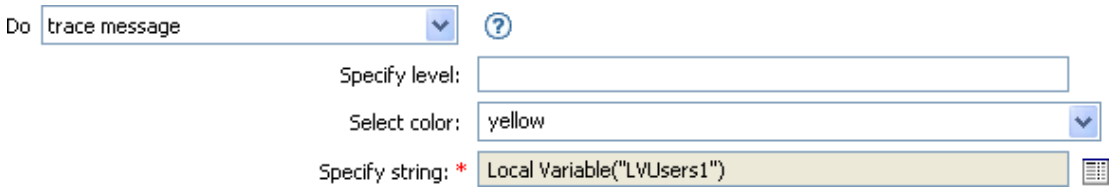
String

Specify the value of the trace message.

Example

The example has four rules that implement a Placement policy for User objects based on the first character of the Surname attribute. It generates both a trace message and a custom Sentinel event. The Trace Message action is used to send a trace message to DSTRACE. The policy name is Policy to Place by Surname and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Placement-BySurname.xml` (../samples/001-Placement-BySurname.xml).





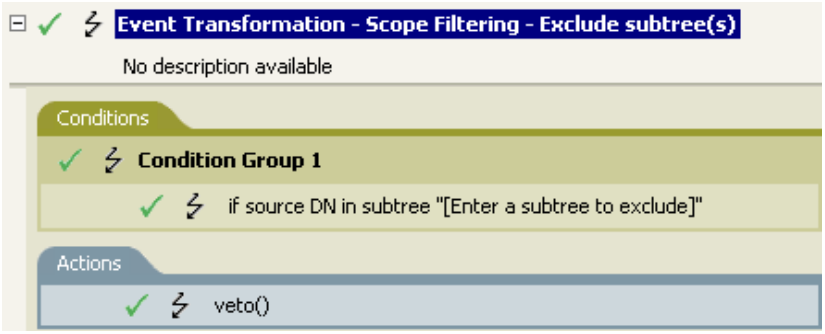
The action sends a trace message to DSTRACE. The contents of the local variable is LVUsers1 and it shows up in yellow in DSTRACE.

Veto

Vetoes the current operation.

Example

The example excludes all events that come from the specified subtree. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Event Transformation - Scope Filtering - Exclude Subtrees" on page 120. To view the policy in XML, see `predef_transformation_filter_exclude_subtress.xml` (../samples/predef_transformation_filter_exclude_subtrees.xml).



The action vetoes all events that come from the specified subtree.

Veto If Operation Attribute Not Available

Conditionally cancels the current operation and ends processing of the current policy, based on the availability of an attribute in the current operation.

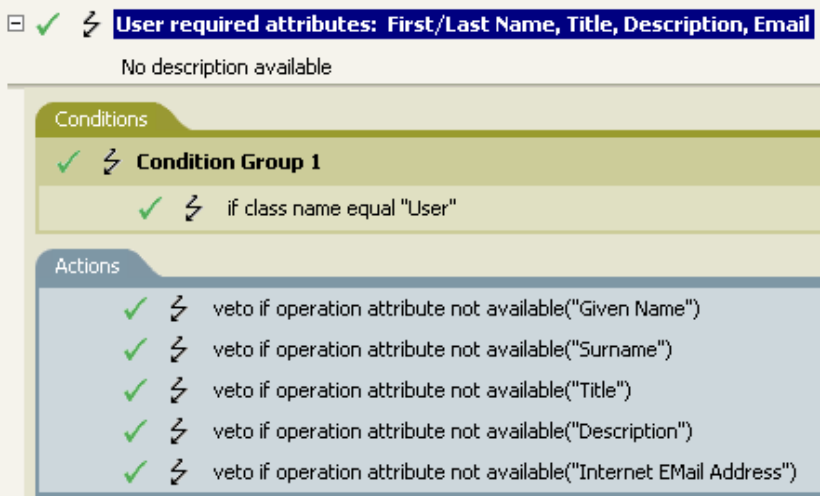
Fields

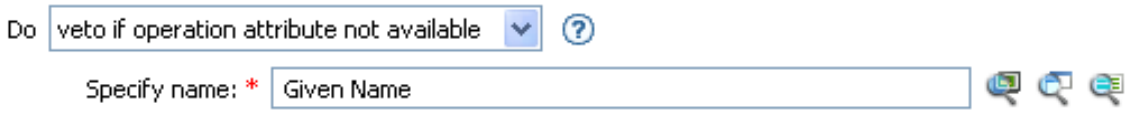
Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

The example does not allow User objects to be created unless the attributes Given Name, Surname, Title, Description, and Internet EMail Address are available. The policy name is Policy to Enforce the Presences of Attributes, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Create-RequiredAttrs.xml` (../samples/001-Create-RequiredAttrs.xml).





The actions vetoes the operation if the attributes of Given Name, Surname, Title, Description, and Internet Email Address are not available.

While

Causes the specified actions to be repeated while the specified conditions evaluate to True.

Fields

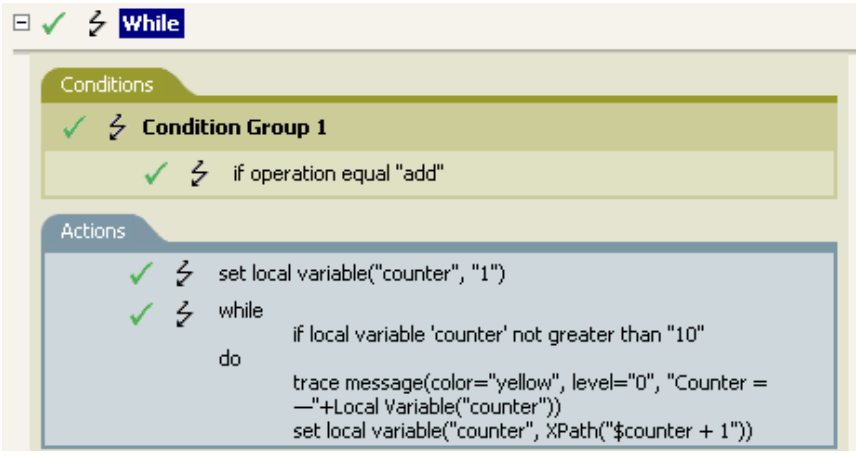
Conditions

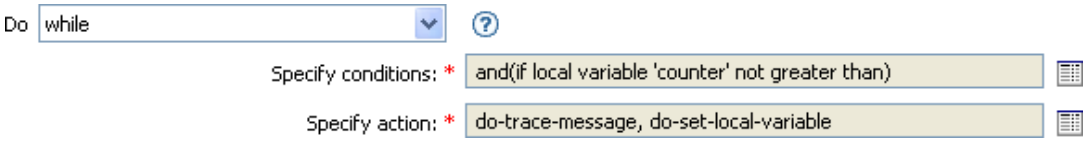
Specify the condition to be evaluated.

Actions

Specify the actions to be repeated if the conditions evaluate to True.

Example





13 Noun Tokens

Noun tokens expand to values that are derived from the current operation, the source or destination data stores, or some external source.

NOTE: For information about elements in the XML schema, see NDS DTD in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

This section contains detailed information about the noun tokens available in the Policy Builder interface.

- "Text" on page 343
- "Added Entitlement" on page 345
- "Association" on page 346
- "Attribute" on page 347
- "Character" on page 348
- "Class Name" on page 349
- "Destination Attribute" on page 350
- "Destination DN" on page 352
- "Destination Name" on page 354
- "Document" on page 355
- "Entitlement" on page 356
- "Generate Password" on page 357
- "Global Configuration Value" on page 358
- "JSON Array" on page 359
- "JSON Object" on page 360
- "Local Variable" on page 361
- "Named Password" on page 362
- "Operation" on page 364
- "Operation Attribute" on page 365
- "Operation Property" on page 367
- "Password" on page 368
- "Query" on page 369
- "Removed Attribute" on page 371
- "Removed Entitlement" on page 372
- "Resolve" on page 373
- "Source Attribute" on page 374

- “Source DN” on page 375
- “Source Name” on page 376
- “Time” on page 377
- “Unique Name” on page 378
- “Unmatched Source DN” on page 381
- “XPath” on page 382

Text

Expands to the text.

Fields

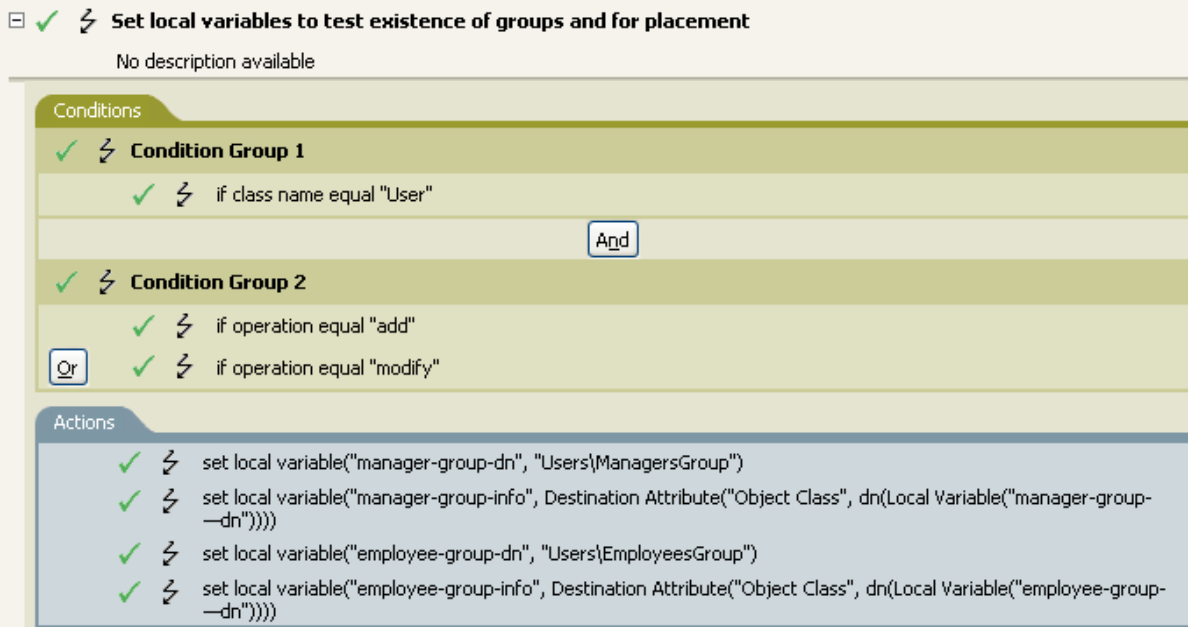
Text

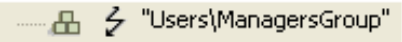
Specify the text. Supports variable expansion. For more information, see "Variable Selector" on page 31.

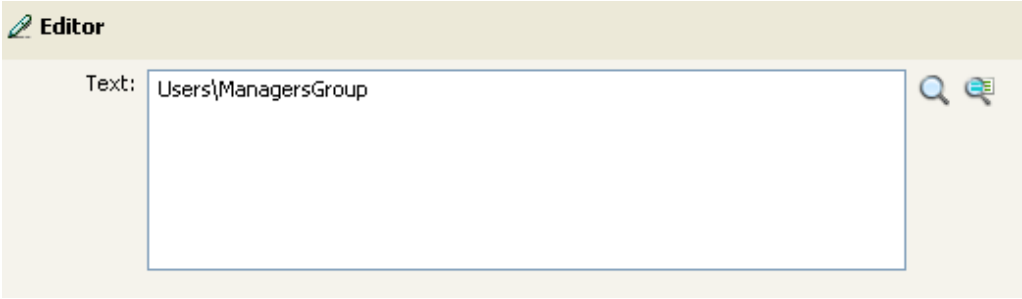
Example

The example is from the Govern Groups for User Based on Title policy, which is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).

The Text token is used in the action Set Location Variable to define the DN of the manager's group. The Text token can contain objects or plain text.







The Text token contains the DN for the manager's group. You can browse to the object you want like to use, or type the information into the editor.

Added Entitlement

Expands to the values of an entitlement granted in the current operation.

Fields

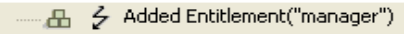
Name

Name of the entitlement. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example



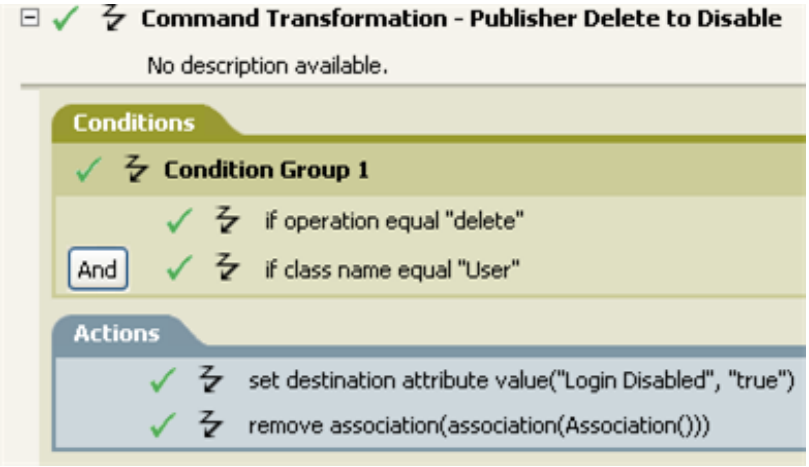
Association

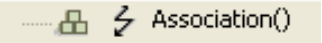
Expands to the association value from the current operation.

Example

The example is from the predefined rules that come with OpenText Identity Manager. For more information on the predefined rule, see "Command Transformation - Publisher Delete to Disable" on page 110.

The action of Remove Association uses the Association token to retrieve the value from the current operation. The rule removes the association from the User object so that any new events coming through do not affect the User object. To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).





Attribute

Expands to the value of an attribute from the current object in the current operation and in the source data store. It can be logically thought of as the union of the operation attribute token and the source attribute token. It does not include the removed values from a Modify operation.

Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

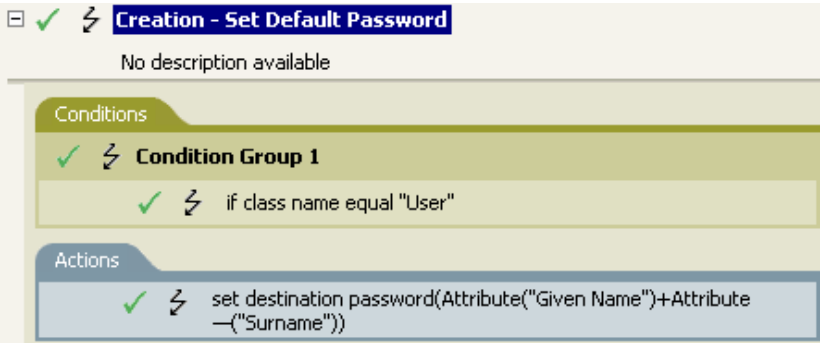
Remarks

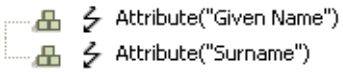
If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

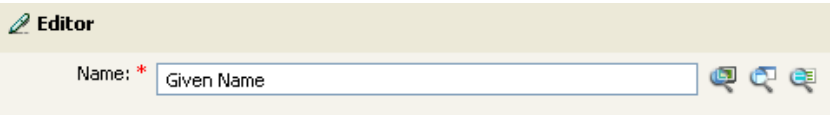
Example

The example is from the predefined rules that come with OpenText Identity Manager. For more information, see "Creation - Set Default Password" on page 117.

The action of Set Destination Password uses the attribute token to create the password. The password is made up of the Given Name attribute and the Surname attribute. When you are in the Argument Builder Editor, you browse and select the attribute you want to use. To view the policy in XML, see `predef_creation_set_default_password.xml` (../samples/predef_creation_set_default_password.xml).







Character

Expands to a character specified by a Unicode* code point.

Remarks

For a listing of Unicode values and characters, see Unicode Code Charts (http://www.unicode.org/charts/).

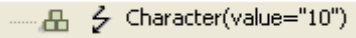
Fields

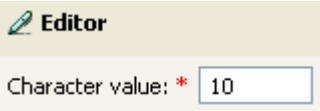
Character Value

The Unicode code point of the character. Supports variable expansion. For more information, see "Variable Selector" on page 31.

A hexadecimal number can be specified if it is prefixed with `0x`, as in C-based programming languages.

Example

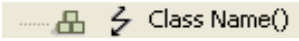




Class Name

Expands to the object class name from the current operation.

Example



Destination Attribute

Expands to the specified attribute value an object.

Fields

Name

Name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Select Object

Select Current Object, Association, or DN.

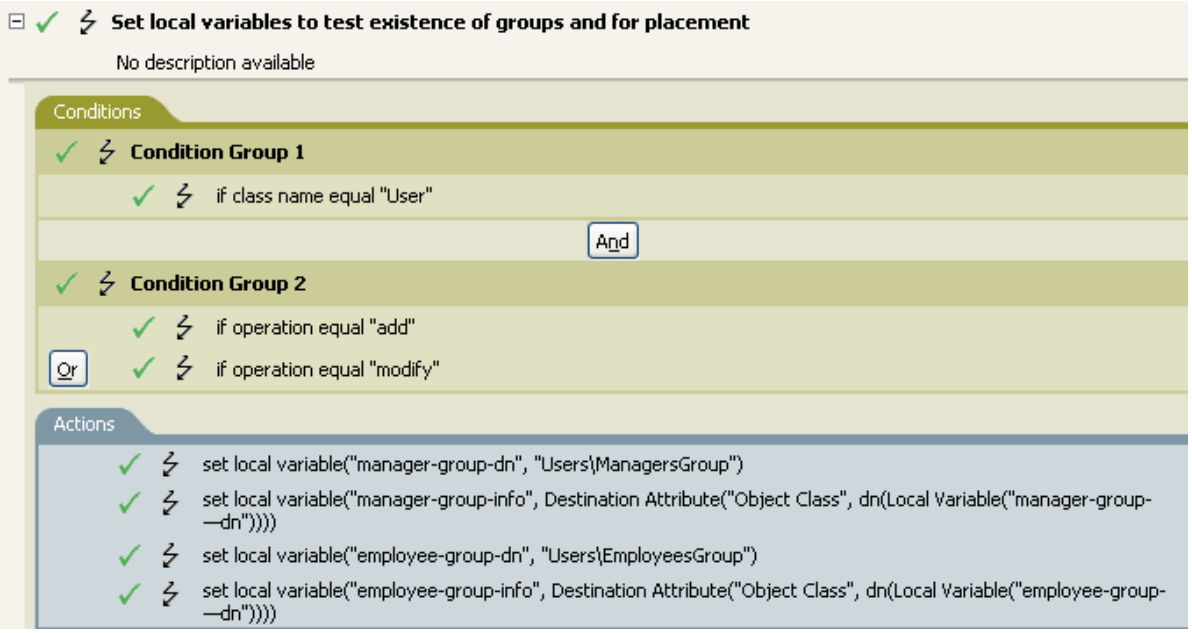
Remarks

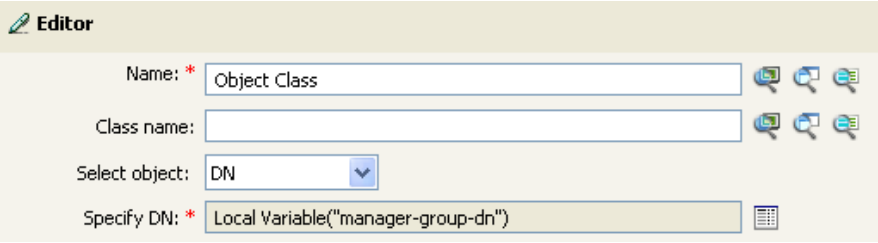
If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

Example

The example is from the Govern Groups for User Based on Title policy, which is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).

The policy creates the Destination Attribute with the Argument Builder. The action of Set Local Variable contains the Destination Attribute token.





You build the Destination Attribute through the Editor. In this example, the attribute of Object Class is set. The DN is used to select the object. The value of DN is the Local Variable of manager-group-dn.

Destination DN

Expands to the destination DN specified in the current operation.

Fields

Start

Specify the RDN index to start with:

- Index 0 is the root-most RDN
- Positive indexes are an offset from the root-most RDN
- Index -1 is the leaf-most segment
- Negative indexes are an offset from the leaf-most RDN towards the root-most RDN

Length

Specify the number of RDN segments to include. Negative numbers are interpreted as (total # of segments + length) + 1. For example, for a DN with 5 segments a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

Convert

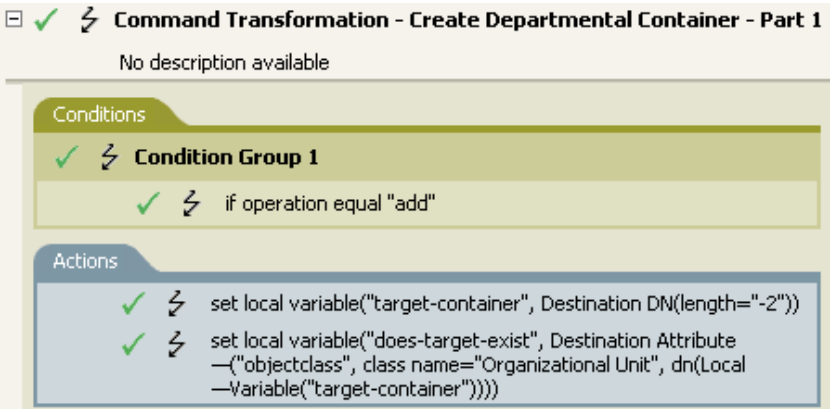
Select whether or not to convert the DN to the format used by the source data store.

Remarks

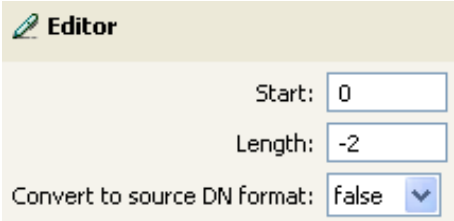
If start and length are set to the default values {0,-1}, the entire DN is used; otherwise, only the portion of the DN specified by start and length is used.

Example

The example uses the Destination DN token to set the value for the local variable of target-container. The policy creates a department container for the User object if it does not exist. The policy is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Create Departmental Container - Part 1 and Part 2" on page 108. To view the policy in XML, see `predef_command_create_dept_container1.xml` (../samples/predef_command_create_dept_container1.xml).



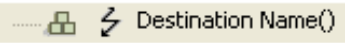
Destination DN(length="-2")


Editor
Start:
0
Length:
-2
Convert to source DN format:
false

Destination Name

Expands to the unqualified Relative Distinguished Name (RDN) of the destination DN specified in the current operation.

Example



Document

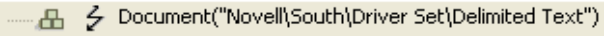
Reads the XML document pointed to by the URI and returns the document node in a node set. The URI can be relative to the URI of the including policy. With any error, the result is an empty node set.

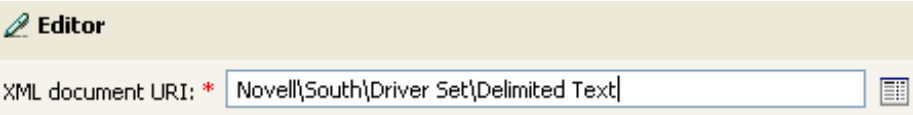
Fields

XML Document URI

Specify the XML document URI.

Example





Entitlement

Expands to the values of a granted entitlement from the current object.

Fields

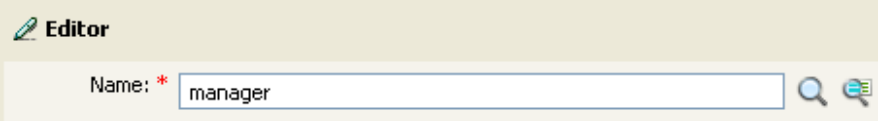
Name

Name of the entitlement. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example



Generate Password

Generates a random password that conforms to the password policy specified by policy-dn. 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 in OpenText eDirectory (for example, the target of an add operation on the publisher channel), the effective password policy of the target container is used.

Fields

Password Policy

The DN of the password policy that receives the randomly generated password. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Set DN relative to policy

Select whether the DN of the password policy is relative to the policy being created.

Example

```
Generate Password(policy-dn="\[root]\Security\Password Policies\Sample Password Policy")
```

Global Configuration Value

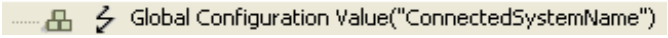
Expands to the value of a global configuration variable.

Fields

Name

Name of the global configuration value. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example



JSON Array

Constructs a JSON array from the enclosed `<arg-strings>`. Each `<arg-string>` is evaluated as an element in the JSON array.

Example

```
<arg-string name="rights">
                <token-json-array>
                   <arg-string>
                                <token-text xml:space="preserve">"admin"</
token-text>
                   </arg-string>
                         <arg-string>
                        <token-text xml:space="preserve">"manager"</token-
text>
                         </arg-string>
                   <arg-string>
                               <token-text xml:space="preserve">"editor"</
token-text>
                   </arg-string>
                </token-json-array>
       </arg-string>
```

The above example is evaluated as the following JSON array.

```
 "rights":["admin","manager","editor"]
```

JSON Object

Constructs a JSON string from the enclosed `<arg-strings>`. Each `<arg-string>` is evaluated as a key-value pair where name in `<arg-string>` is the key and the enclosed text is evaluated as the value. JSON Arrays can be defined in a JSON Object. The below example has a JSON array defined in a JSON object.

Example

```
<token-json-object>
         <arg-string name="name">
             <token-text xml:space="preserve">"Mike"</token-text>
         </arg-string>
       <arg-string name="age">
                 <token-text xml:space="preserve">40</token-text>
       </arg-string>
         <arg-string name="disabled">
             <token-text xml:space="preserve">false</token-text>
         </arg-string>
       <arg-string name="rights">
                <token-json-array>
                   <arg-string>
                                <token-text xml:space="preserve">"admin"</
token-text>
                   </arg-string>
                         <arg-string>
                        <token-text xml:space="preserve">"manager"</token-
text>
                         </arg-string>
                   <arg-string>
                             <token-text xml:space="preserve">"reviewer"</
token-text>
                   </arg-string>
                </token-json-array>
       </arg-string>
</token-json-object>
The above example is evaluated as the following JSON string.

{"name":"Mike","age":40,"disabled":false,"rights":["admin","manager","revi
ewer"]}
```

Local Variable

Expands to the value of a local variable.

Fields

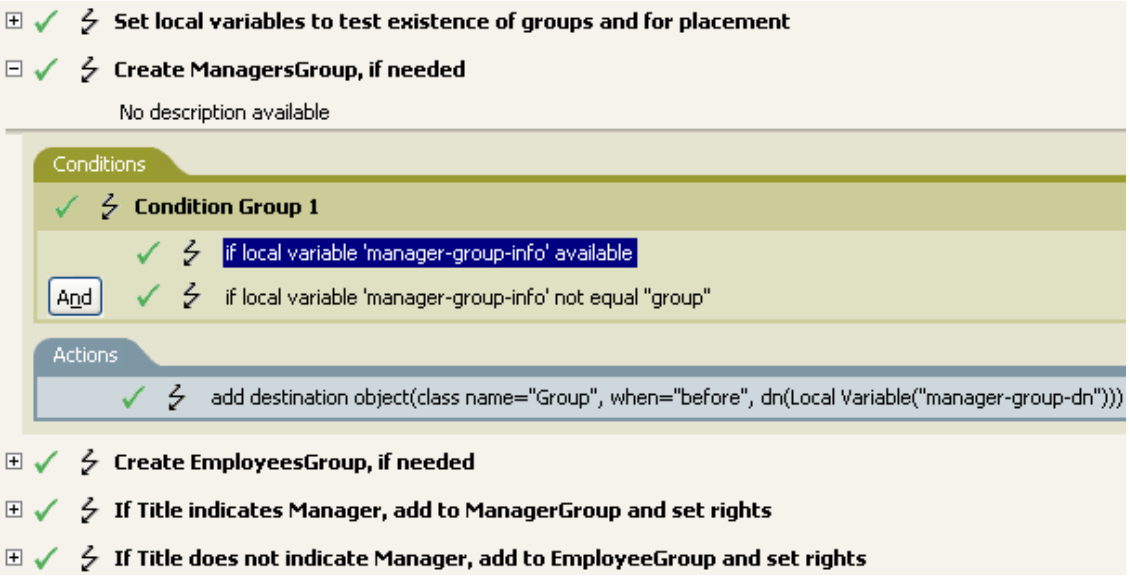
Name

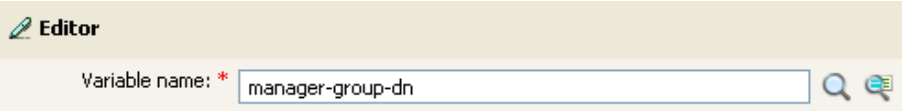
Specify the name of the local variable. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

The example is from the Govern Groups for User Based on Title policy, which is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `003-Command-AddCreateGroups.xml` (../samples/003-Command-AddCreateGroups.xml).

The action Add Destination Object uses the Local Variable token.





The Local Variable can only be used if the action Set Local Variable has been used previously in the policy. It sets the value that is stored in the Local Variable. In the Editor, you click the **browse** icon and all of the local variables that have been defined are listed. Select the correct local variable.

The value of the local variable is group-manager-dn. In the example, the Set Local Variable action defines group-manager-dn as DN of the manager's group Users\ManagersGroup.

Named Password

Expands to the named password from the driver.

Fields

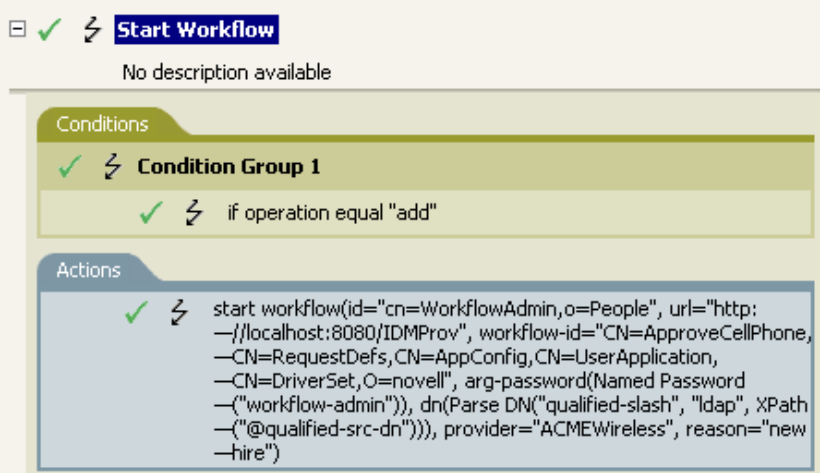
Name

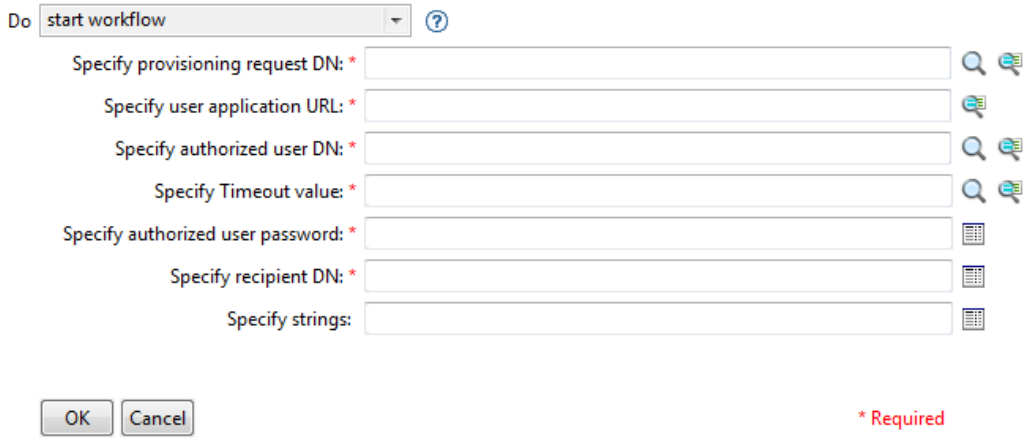
Name of the password. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

The Named Password noun token can only be used if a Named Password has been set on the driver object. The Named Password is used to save a password in an encrypted form. For more information on Named Passwords, see "Securely Storing Driver Passwords with Named Passwords" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

The example uses the Start Workflow (page 330) action. It requires that the password for the workflow administrator be entered. To view the policy in XML, see `start_workflow.xml` (../samples/start_workflow.xml).





Named Password("workflow-admin")

Editor

Password name: * workflow-admin

Select Named Password

The selected named password is passed to the expression in the Argument Builder.

Server:

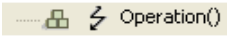
server

Name	Display Name
smpt-admin	smpt-admin
workflow-admin	workflow-admin

Operation

Expands to the name of the current operation.

Example



Operation Attribute

Expands to the value of an attribute in the current operation. The operation can be an `<add-attr>`, `<add-value>`, or `<attr>`. If this token is evaluated in a context where a node-set result is expected, then all the available values are returned as nodes in a node-set. Otherwise, the first available value is returned as a string.

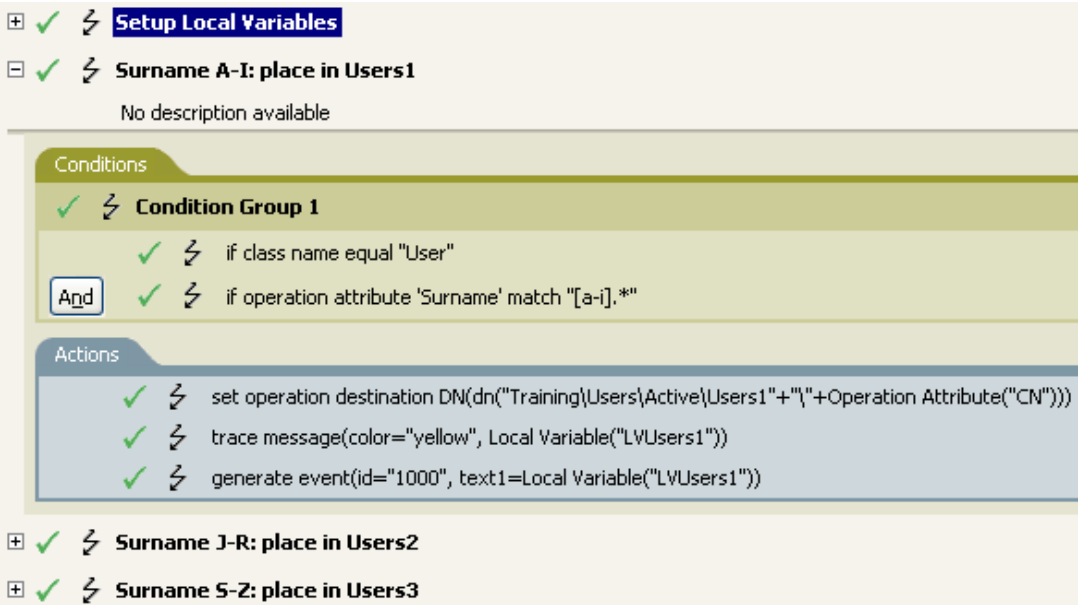
Fields

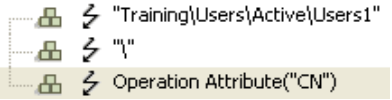
Name

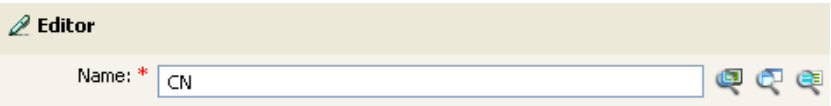
Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

The example has four rules that implement a Placement policy for User objects based on the first character of the Surname attribute. It generates both a trace message and a custom Sentinel event. The policy name is Policy to Place by Surname, and it is available for download from the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Placement-BySurname.xml` (../samples/001-Placement-BySurname.xml).







The action Set Operation Destination DN contains the Operation Attribute token. The Operation Attribute token sets the Destination DN to the CN attribute. The rule takes the context of Training\Users\Active\Users and adds a \ plus the value of the CN attribute.

Operation Property

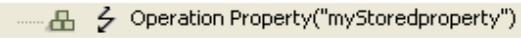
Expands to the value of the specified operation property on the current operation.

Fields

Name

Specify the name of the operation property. Supports variable expansion. For more information, see "Variable Selector" on page 31.

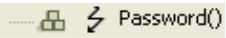
Example



Password

Expands to the password specified in the current operation.

Example



Query

Queries the source or destination data store and returns the resulting instances.

Fields

Datastore

Specify the data store to query.

Scope

Select the scope of the query. The options are entry, subordinates, or subtree.

Max Result Count

Specify the maximum number of results returned from the query.

Class Name

Specify the class name in the query. If a class name is not specified, all classes are searched. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Select Object

Specify the base of the query. It can be the DN or an association. If neither is selected, the query starts at the root of the datastore.

Match Attributes

Select the attributes to search for.

IMPORTANT: To improve performance when using the query noun, create an index for the attributes that you are going to use when querying the Identity Vault. For more information about indexes, see the OpenText eDirectory Administration Guide (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/bookinfo.html).

Read Attribute

Specify the set of attributes to return. If nothing is specified, no attributes are read. Use an asterisk to read all attributes.

Remarks

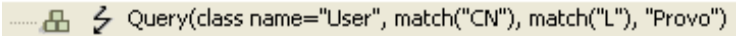
The Query token returns a `node set` containing the instance elements found as a result of the query. To effectively use the results of a Query token it must be used in a context that is expecting a `node set`. For example, you could assign the result to a variable of type `node set`, or iterate through the result using a `for each` loop.

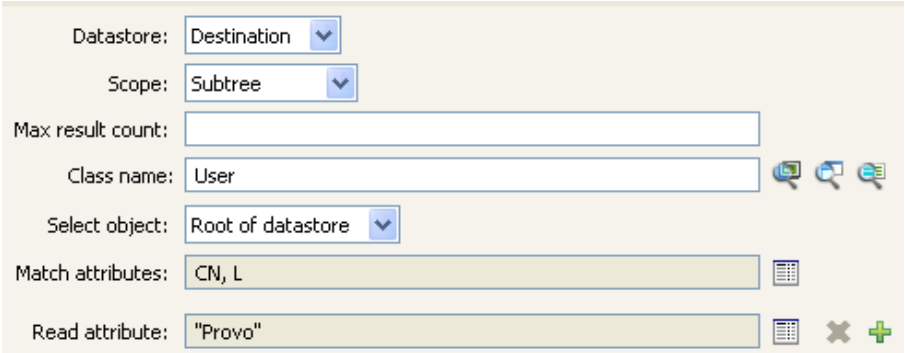
Treating the `node set` as if it were a `string` seldom provides anything useful. Extracting useful information from the `node set` or its constituent instance elements requires the use of an XPath expression and knowledge of the structure of an instance element. For additional information, see the following:

- **instance** in the *OpenText Identity Manager DTD Reference Documentation (https://www.netiq.com/documentation/identity-manager-developer/dtd-documentation.html)*.

- "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.
- Chapter 4, "Using the XPath Builder," on page 69.

Example





Removed Attribute

Expands to the specified attribute value being removed in the current operation. It applies only to a Modify operation.

Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

Example

Removed Entitlement

Expands to the values of the an entitlement revoked in the current operation.

Fields

Name

Specify the name of the entitlement. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that entitlement. If it is used in a context where a string is expected, the token expands to the string value found.

Example

Resolve

Resolves the DN to an association key, or the association key to a DN in the specified data store.

Fields

Datastore

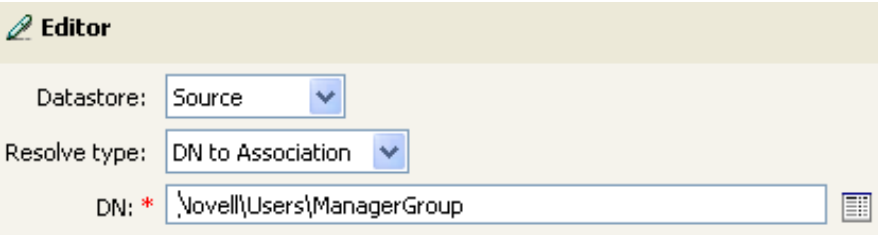
Select the destination or source datastore to be queried.

Resolve Type

Select to resolve the association key to a DN or to resolve the DN to an association key.

Example

```
Resolve(datastore="src", dn())
```



Source Attribute

Expands to the values of an attribute from an object in the source data store.

Fields

Name

Name of the attribute. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object. Supports variable expansion. For more information, see "Variable Selector" on page 31.

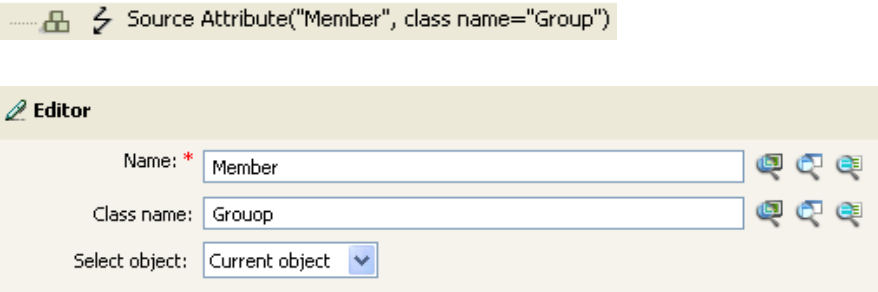
Object

Select the source object. This object can be the current object, or can be specified by a DN or an association.

Remarks

If the token is used in a context where a node set is expected, the token expands to a node set containing all of the values for that attribute. If it is used in a context where a string is expected, the token expands to the string value found.

Example



Source DN

Expands to the source DN from the current operation.

Fields

Start

Specify the RDN index to start with:

- Index 0 is the root-most RDN
- Positive indexes are an offset from the root-most RDN
- Index -1 is the leaf-most segment
- Negative indexes are an offset from the leaf-most RDN towards the root-most RDN

Length

Number of RDN segments to include. Negative numbers are interpreted as (total # of segments + length) + 1. For example, for a DN with 5 segments a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

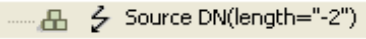
Convert

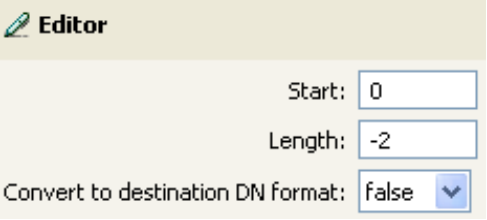
Select whether or not to convert the DN to the format used by the destination data store.

Remarks

If start and length are set to the default values {0,-1}, the entire DN is used; otherwise, only the portion of the DN specified by start and length is used.

Example

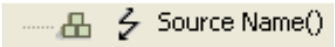




Source Name

Expands to the unqualified relative distinguished name (RDN) of the source DN specified in the current operation.

Example



Time

Expands to the current date/time into the format, language, and time zone specified.

Fields

Format

Specify the date/time format. Select a named time format or specify a custom format pattern. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Language

Specify the language. (It defaults to the current system language.) Supports variable expansion. For more information, see "Variable Selector" on page 31.

Time zone

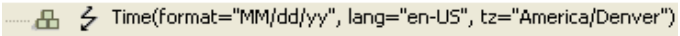
Specify the time zone. (It defaults to the current system time zone.) Supports variable expansion. For more information, see "Variable Selector" on page 31.

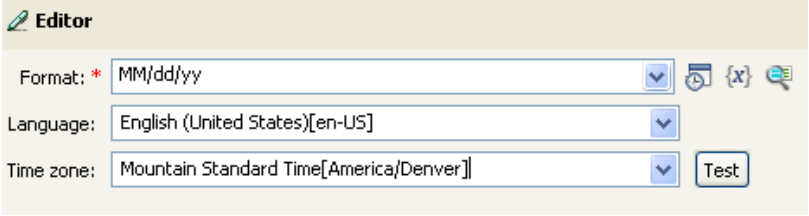
Remark

The **Test** icon displays the time format that is created by selecting the format, language, and time zone.

NOTE: If you want to display additional characters in a format pattern string, you can include those characters by enclosing them in single quotes. For example, you could specify the format pattern `MM/dd/yy'-Local Time'`.

Example





Unique Name

Expands to a pattern-based name that is unique in the destination data store according to the criteria specified.

Fields

Attribute Name

Specify the name of attribute to check for uniqueness.

IMPORTANT: To improve performance when using the unique name noun, create an index for the attributes that you are going to use when querying the Identity Vault. For more information about indexes, see the OpenText eDirectory Administration Guide (https://www.netiq.com/documentation/edirectory-93/edir_admin/data/a5tuuu5.htm).

Scope

Specify the scope in which to check uniqueness. The options are subtree or subordinates.

Start Search

Select a starting point for the search. The starting point can be the root of the data store, or can be specified by a DN or association.

Pattern

Specify patterns to use to generate unique values by using the Argument Builder.

Counters Use

Select when to use a counter. The options are:

- never
- always
- fallback

Counters Pattern

Select which pattern to use the counter with. The options are:

- first
- last
- all

Start

The starting value of the counter. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Digits

Specify the width in digits of counter; the default is 1. The **Pad counter with leading 0's** option prepends 0 to match the digit length. For example, with a digit width of 3, the initial unique value would be appended with 001, then 002, and so on. Supports variable expansion. For more information, see "Variable Selector" on page 31.

If Cannot Construct Name

Select the action to take if a unique name cannot be constructed. The options are:

- Ignore, return empty
- Generate warning, return empty name
- Generate error, abort current transaction
- Generate fatal error, shut down driver

Test all objects

If set to true, the object class is omitted from the search. Otherwise, the operation's object class is used. By default, this attribute is set to false.

Remarks

Each `<arg-string>` element provides a pattern to be used to create a proposed name.

A proposed name is tested by performing a query for that value in the name attribute against the destination data store by using the `<arg-dn>` element or the `<arg-association>` element as the base of the query and scope as the scope of the query. If the destination data store is the Identity Vault and name is omitted, then a search is performed against the pseudo-attribute “[Entry].rdn”, which represents the RDN of an object without respect to what the naming attribute might be. If the destination data store is the application, then name is required.

A pattern can be tested with or without a counter as indicated by counter-use and counter-pattern. When a pattern is tested with a counter, the pattern is tested repeatedly with an appended counter until a name is found that does not return any instances or the counter is exhausted. The counter starting value is specified by counter-start and the counter maximum value is specified in terms of the maximum number of digits as specified by counter-digits. If the number of digits is less than those specified, then the counter is right-padded with zeros unless the counter-pad attribute is set to False. The counter is considered exhausted when the counter can no longer be represented by the specified number of digits.

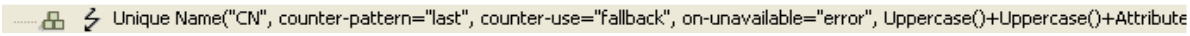
As soon as a proposed name is determined to be unique, the testing of names is stopped and the unique name is returned.

The order of proposed names is tested as follows:

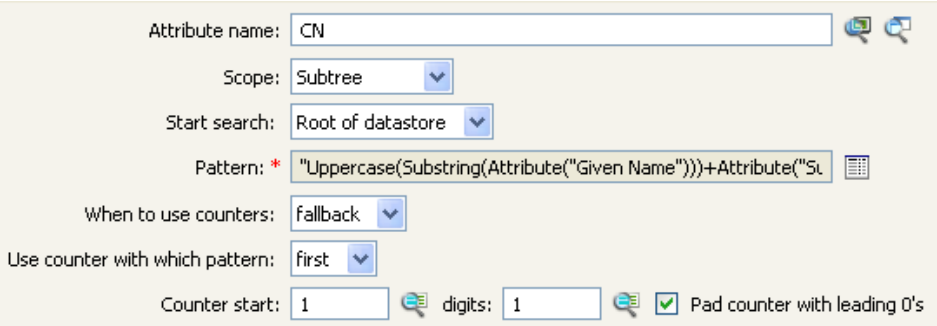
- Each pattern is tested in the order specified. If counter-use=“always” and the pattern is one of the patterns indicated by the counter-pattern, then the pattern is tested with a counter; otherwise, it is tested without a counter.
- If no unique name has been found after the patterns have been exhausted and counter-use=“fallback”, then the patterns indicated by the counter-pattern are retried with a counter.

If all specified combinations of patterns and counters are exhausted, then the action specified by the on-unavailable is taken.

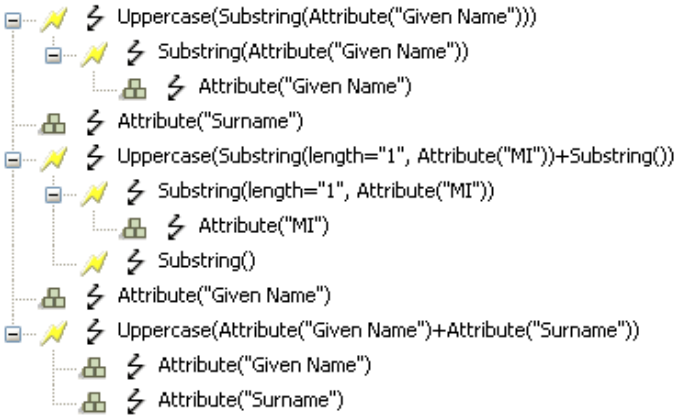
Example



The following is an example of the Editor pane when constructing the unique name argument:



The following pattern was constructed to provide unique names:



If this pattern does not generate a unique name, a digit is appended, incrementing up to the specified number of digits. In this example, nine additional unique names would be generated by the appended digit before an error occurs (pattern1 - pattern99).

Unmatched Source DN

Expands to the part of the source DN in the current operation that corresponds to the part of the DN that was not matched by the most recent match of an If Source DN condition.

Fields

Convert

Select whether or not to convert the DN format used by the destination data store.

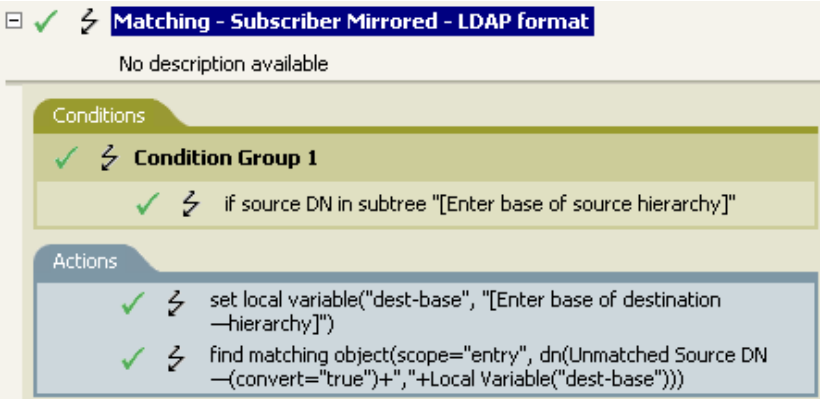
Remarks

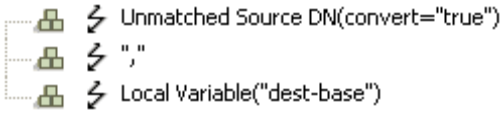
If there are no matches, the entire DN is used.

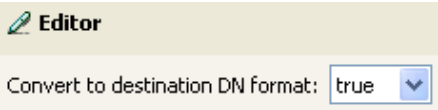
Example

The example is from the predefined rules that come with OpenText Identity Manager. For more information, see "Placement - Subscriber Mirrored - LDAP Format" on page 133. To view the policy in XML, see `predef_match_sub_mirrored.xml` (../samples/predef_match_sub_mirrored.xml).

The action of Finding Matching Object uses the Unmatched Source DN token to build the matching information in LDAP format. It takes the unmatched portion of the source DN to make a match.







XPath

Expands to the results of evaluating an XPath 1.0 expression.

Fields

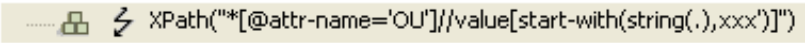
Expression

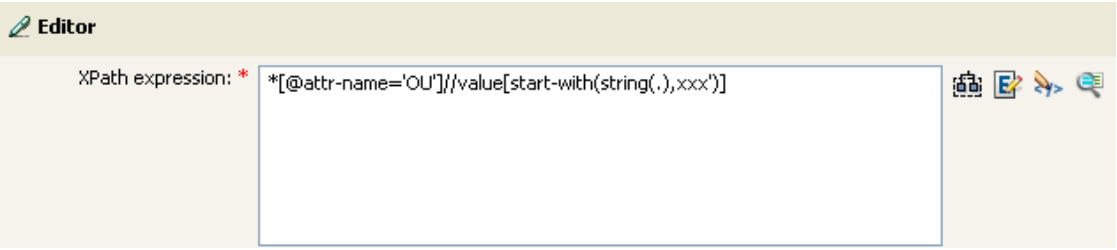
XPath 1.0 expression to evaluate.

Remarks

For more information on using XPath expressions with policies, see "XPath 1.0 Expressions" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*.

Example





14 Verb Tokens

Verb tokens modify the concatenated results of other tokens that are subordinate to them.

This section contains detailed information about all verbs that are available through the Policy Builder interface.

NOTE: For the tokens that support regular expression, OpenText Identity Manager evaluates the following special characters in the regular expression context:

\ $ ^.? * + [ ] ( ) |

To use these characters as literals in a regular expression, escape the character with a backslash ("\").

- "Base64 Decode" on page 384
- "Base64 Encode" on page 385
- "Convert Time" on page 386
- "Escape Destination DN" on page 388
- "Escape Source DN" on page 389
- "Join" on page 390
- "Lowercase" on page 391
- "Map" on page 392
- "Parse DN" on page 394
- "Replace All" on page 396
- "Replace First" on page 397
- "Split" on page 399
- "Substring" on page 400
- "Uppercase" on page 402
- "XML Parse" on page 403
- "XML Serialize" on page 404

Base64 Decode

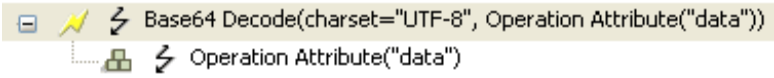
Decodes the result of the enclosed tokens from Base64-encoded data to bytes, then converts the bytes into a string by using the specified character set.

Fields

Character Set

Specify the character set that converts the decoded bytes to a string. It can be any character set supported by Java. If the field is left blank, the character set defaults to the system encoding as specified by the file.encoding System property. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example



Base64 Encode

Converts the result of the enclosed tokens to bytes by using the specified character set, then Base64-encodes the bytes.

Fields

Character Set

Specify the character set that converts the string to bytes. It can be any Java-supported character set. If the filed is left blank, the character set defaults to the system encoding as specified by the file.encoding System property. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Example

```
Base64 Encode(charset="UTF-8", Operation Attribute("Surname"))
    Operation Attribute("Surname")
```

Convert Time

Converts the date and time represented by the result of the enclosed tokens from the source format, language, and time zone to the destination format, language, and time zone.

Fields

Source Format

Specify the source date/time format. Select a named time format or specify a custom format pattern. Supports variable expansion. For more information, see "Variable Selector" on page 31.

NOTE: From Identity Manager 4.8.3 onwards, the `setLenient` flag is set to `False`. Therefore, you cannot specify invalid values for date and time.

Source Language

Specify the source language (defaults to the current system language). Supports variable expansion. For more information, see "Variable Selector" on page 31.

Source Time Zone

Specify the source time zone (defaults to the current system time zone). Supports variable expansion. For more information, see "Variable Selector" on page 31.

Destination Format

Specify the destination date/time format. Select a named time format or specify a custom format pattern. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Destination Language

Specify the destination language (defaults to the current system language). Supports variable expansion. For more information, see "Variable Selector" on page 31.

Destination Time Zone

Specify the destination time zone (defaults to the current system time zone). Supports variable expansion. For more information, see "Variable Selector" on page 31.

Offset

Specifies an offset to apply to the time in the selected noun. Specify an offset number, then select the appropriate time unit from the drop-down list (seconds, minutes, hours, days, weeks, months, years.)

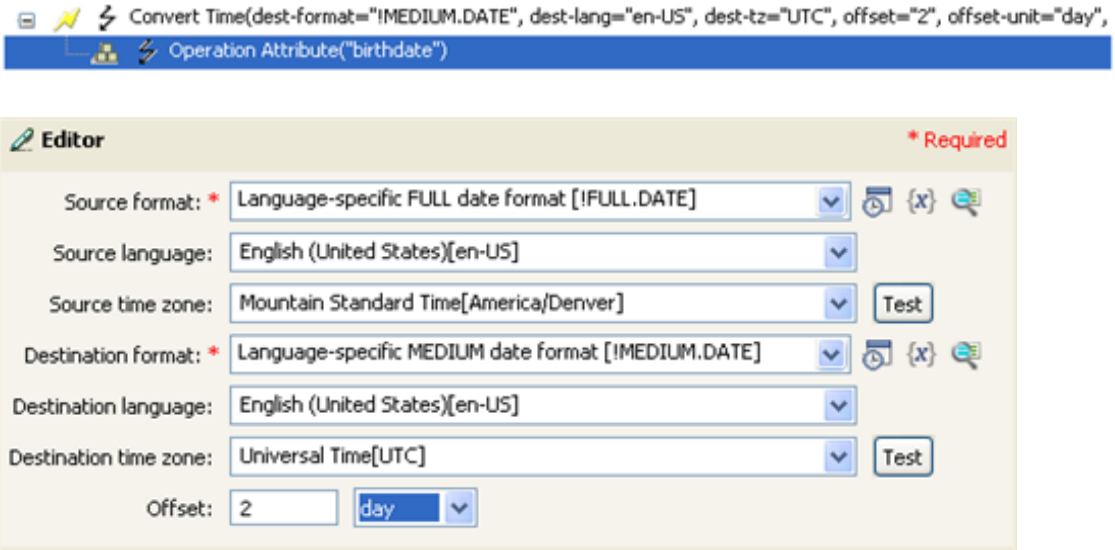
NOTE: ◆This field is only available if the OpenText Identity Manager server version is set to 3.6.

- If you specify a negative offset value, the time is offset to a previous time or date. For example, if you specify `-2` and select **Day**, the time in the selected noun is adjusted to two days prior to the actual date.

Remark

The **Test** icon displays the time format that is created by selecting the format, language, and time zone.

Example



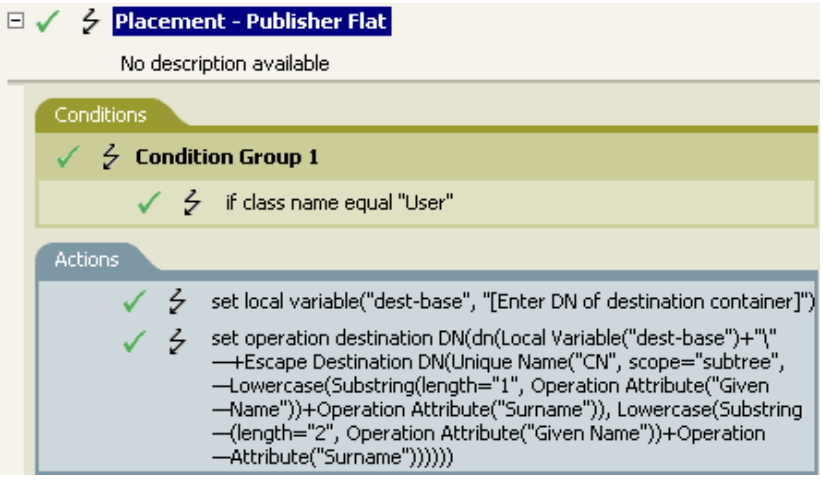
Escape Destination DN

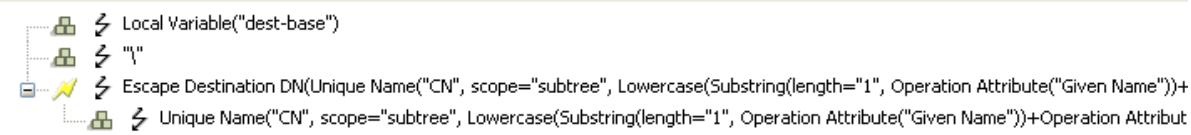
Escapes the enclosed tokens according to the rules of the DN format of the destination data store.

Example

The example is from the predefined rules that come with OpenText Identity Manager. For more information, see "Placement - Publisher Flat" on page 134. To view the policy in XML, see `predef_place_pub_flat.xml` (../samples/predef_place_pub_flat.xml).

The action of Set Operation Destination DN uses the Escape Destination DN token to build the destination DN of the User object.



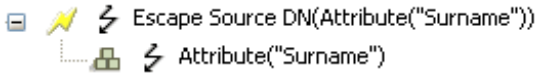


The Escape Destination DN token takes the value in Unique Name and sets it to the format for the destination DN.

Escape Source DN

Escapes the enclosed tokens according to the rules of the DN format of the source data store.

Example



Join

Joins the values of the nodes in the node set result of the enclosed tokens, separating the values by the characters specified by delimiter. If the comma-separated values (CSV) are true, then CSV quoting rules are applied to the values.

Fields

Delimiter

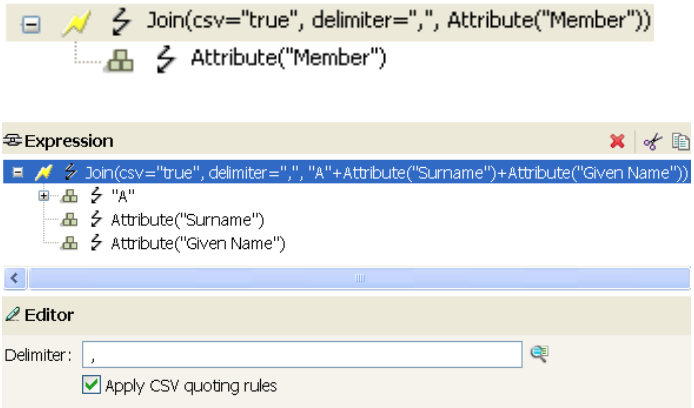
(Optional) Specify the string used to delimit the joined values. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Apply CSV Quoting Rules

Applies CSV quoting values.

Example

The example combines all of the members of the group into a CSV record.

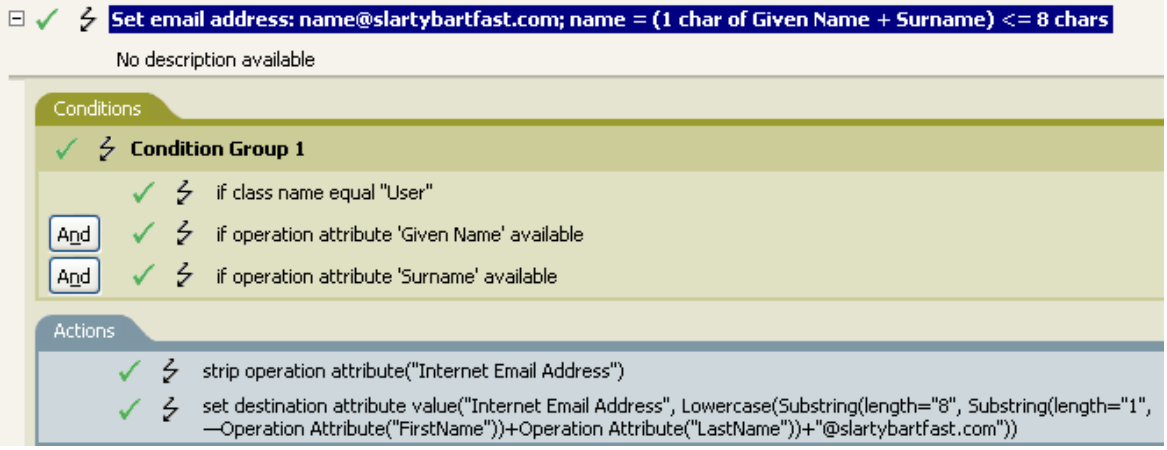


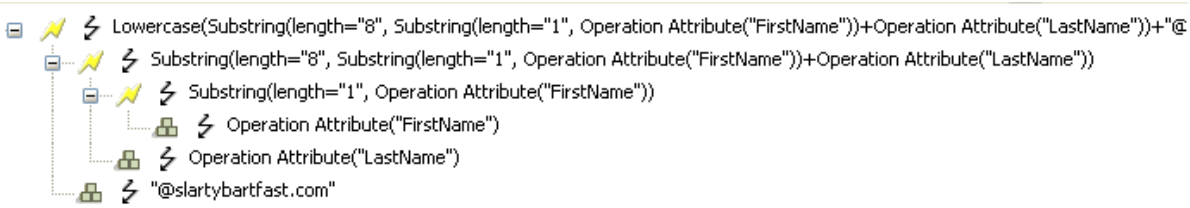
Lowercase

Converts the characters in the enclosed tokens to lowercase.

Example

This example sets the e-mail address to be name@slartybartfast.com where the name equals the first character of the Given Name plus the Surname. The policy name is Policy: Create E-mail from Given Name and Surname, and it is available for download at the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see 001-Command-`SetEmailByGivenNameAndSurnam.xml` (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).





The Lowercase token sets all of the information in the action Set Destination attribute value to lowercase.

Map

Maps the result of the enclosed tokens from the values specified by the source column to the destination column in the specified mapping table.

Remarks

If this token is evaluated in a context where a node set result is expected and multiple rows are matched by the value being mapped, a node set is returned that contains the values from the destination column of each matching row. Otherwise, only the value from the first matching row is returned.

The table attribute should be the slash form DN of the Resource object containing the mapping table to be used. The DN might be relative to the including policy.

Fields

Mapping Table DN

Specify the slash form DN of a Resource object containing the mapping table. Supports variable expansion. You can click Choose the Mapping table icon to select a mapping table from the Model Browser window. For more information, see "Variable Selector" on page 31.

Set DN Relative to Policy

When it is enabled, it displays only the mapping table DN relative to the policy. This is the default.

Source Column Name

Specify the name of the source column. Supports variable expansion. You can also select the Source Column Names that are pre-defined. For more information, see "Variable Selector" on page 31.

NOTE: From Identity Manager 4.8 onwards, the `src` attribute is modified to `source`.

Destination Column Name

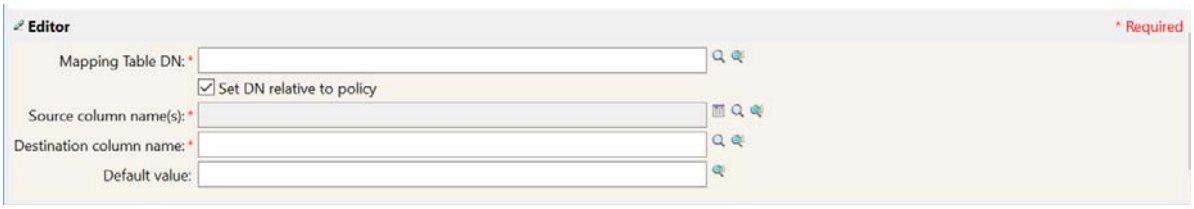
Specify the name of the destination column. Supports variable expansion. You can also select the Destination Column Names that are pre-defined. For more information, see "Variable Selector" on page 31.

Default Value

Specifies a value to return if the value being mapped does not match any values in the Source column. You can chose a default value using Launch variable browser icon.

NOTE: This field is only available if the OpenText Identity Manager server version is set to 3.6.

Example


Editor
* Required
Mapping Table DN: *
Set DN relative to policy
Source column name(s): *
Destination column name: *
Default value:

Parse DN

Converts the enclosed token's DN to an alternate format. However, you can not convert a non-qualified DN format to qualified DN using this token.

Fields

Start

Specify the RDN index to start with:

- Index 0 is the root-most RDN
- Positive indexes are an offset from the root-most RDN
- Index -1 is the leaf-most segment
- Negative indexes are an offset from the leaf-most RDN towards the root-most RDN

Length

Number of RDN segments to include. Negative numbers are interpreted as (total # of segments + length) + 1. For example, for a DN with 5 segments a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

Source DN Format

Specifies the format used to parse the source DN.

Destination DN Format

Specify the format used to output the parsed DN.

Source DN Delimiter

Specify the custom source DN delimiter set if Source DN Format is set to custom.

Destination DN Delimiter

Specify the custom destination DN delimiter set if Destination DN Format is set to custom.

Remarks

If start and length are set to the default values {0,-1}, then the entire DN is used; otherwise, only the portion of the DN specified by start and length is used.

When specifying custom DN formats, the eight characters that make up the delimiter set are defined as follows:

- Typed Name Boolean Flag: 0 means names are not typed, and 1 means names are typed
- Unicode No-Map Character Boolean Flag: 0 means don't output or interpret unmappable Unicode characters as escaped hex digit strings, such as \FEFF. The following Unicode characters are not accepted by OpenText eDirectory: 0xfeff, 0xfffe, 0xfffd, and 0xffff.
- Relative RDN Delimiter
- RDN Delimiter
- Name Divider
- Name Value Delimiter

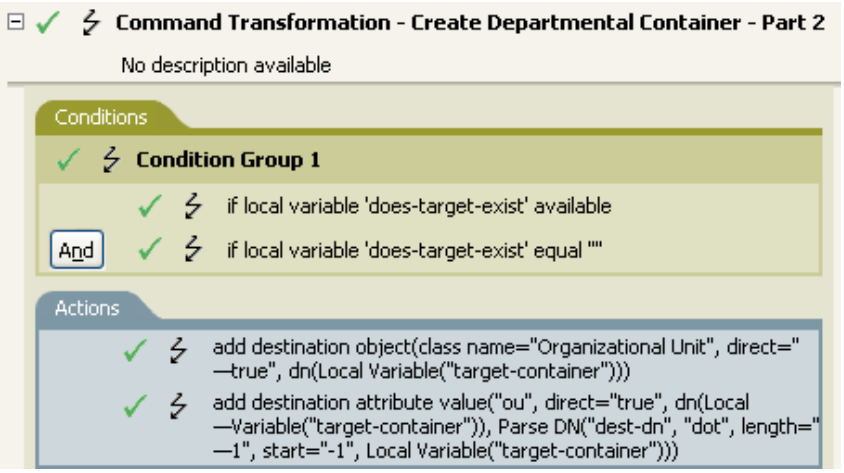
- Wildcard Character
- Escape Character

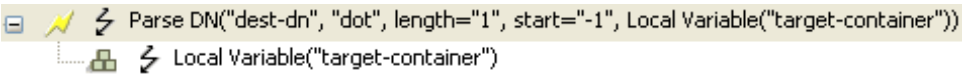
If RDN Delimiter and Relative RDN Delimiter are the same character, the orientation of the name is root right, otherwise the orientation is root left.

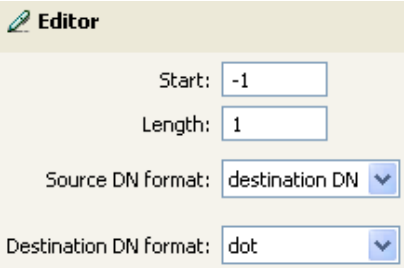
If there are more than eight characters in the delimiter set, the extra characters are considered as characters that need to be escaped, but they have no other special meaning.

Example

The example uses the Parse DN token to build the value the Add Destination Attribute Value action. The example is from the predefined rules that come with OpenText Identity Manager. For more information, see "Command Transformation - Create Departmental Container - Part 1 and Part 2" on page 108. To view the policy in XML, see `predef_command_create_dept_container2.xml` (../samples/predef_command_create_dept_container2.xml).







The Parse DN token takes the information from the source DN and converts it to dot notation. The information from the Parse DN is stored in the attribute value of OU.

Replace All

Replaces all occurrences of a regular expression in the enclosed tokens.

Fields

Regular Expression

Specify the regular expression that matches the substring to be replaced. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Replace With

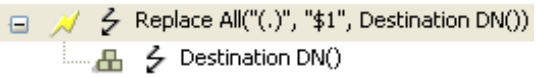
Specify the replacement string. Supports variable expansion. For more information, see "Variable Selector" on page 31.

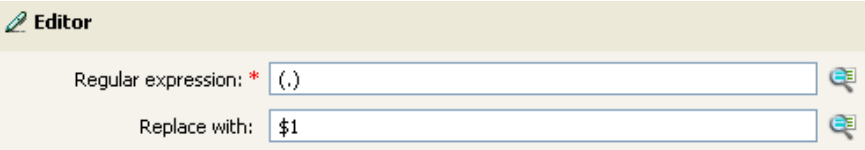
Remarks

For information about creating regular expressions, see Java Class Pattern and Java Class Matcher details for your version of Java in the Oracle Java documentation.

The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.

Example





Replace First

Replaces the first occurrence of a regular expression in the enclosed tokens.

Fields

Regular Expression

Specify the regular expression that matches the substring to replace. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Replace With

Specify the replacement string. Supports variable expansion. For more information, see "Variable Selector" on page 31.

Remarks

The matching instance is replaced by the string specified in the **Replace with field**.

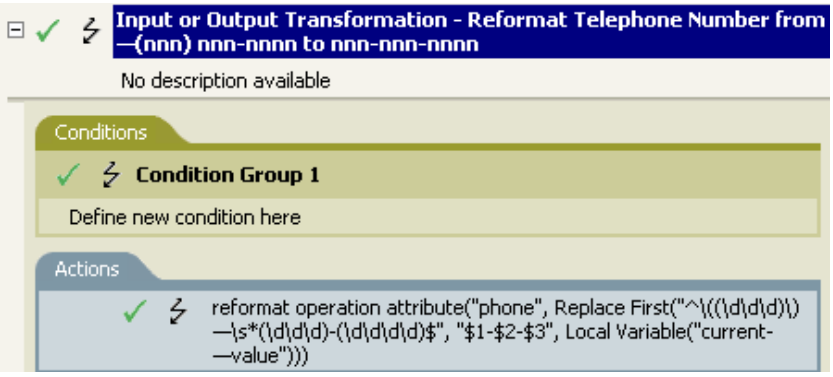
For information about creating regular expressions, see Java Class Pattern and Java Class Matcher details for your version of Java in the Oracle Java documentation.

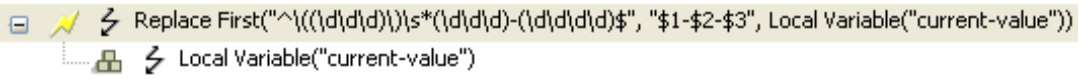
The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.

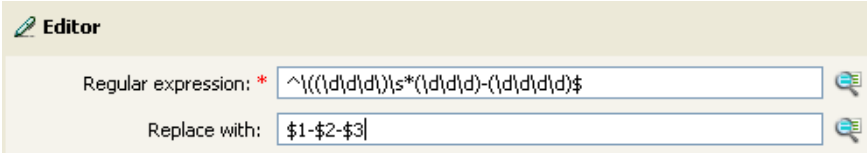
Example

The example reformats the telephone number (nnn)-nnn-nnnn to nnn-nnn-nnnn. The rule is from the predefined rules that come with OpenText Identity Manager. For more information, see "Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn" on page 122. To view the policy in XML, see `predef_transformation_reformat_telephone1.xml` (../samples/predef_transformation_reformat_telephone1.xml).

The Replace First token is used in the Reformat Operation Attribute action.







The regular expression of ^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$ represents (nnn) nnn-nnnn and the regular expression of $1-$2-$3 represents nnn. This rule transforms the format of the telephone number from (nnn) nnn-nnnn to nnn-nnn-nnnn.

Split

Splits the result of the enclosed tokens into a node set consisting of text nodes based on the pattern specified by delimiter. If comma-separated values (CSV) are true, then CSV quoting rules are honored during the parsing of the string.

Fields

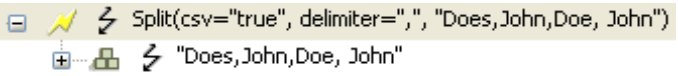
Delimiter

Regular expression that matches the delimiter characters. Supports variable expansion. For more information, see “Variable Selector” on page 31.

Apply CSV Quoting Rules

Applies CSV quoting values.

Example



Substring

Extracts a portion of the enclosed tokens.

Fields

Start

Specify the starting character index:

- Index 0 is the first character.
- Positive indexes are an offset from the start of the string.
- Index -1 is the last character.
- Negative indexes are an offset from the last character toward the start of the string.

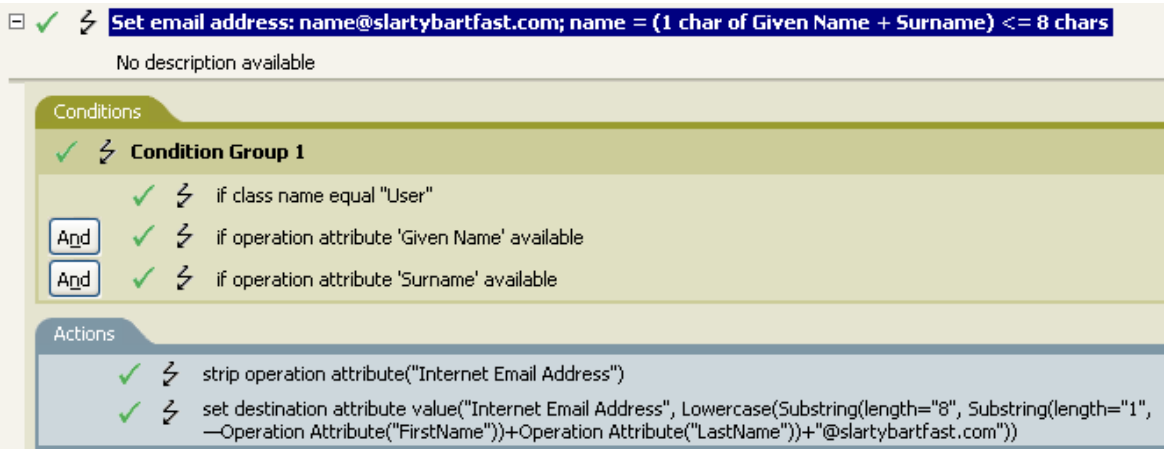
For example, if the start is specified as -2, then it starts reading at the first character from the end. If -3 is specified, then it starts 2 characters from the end.

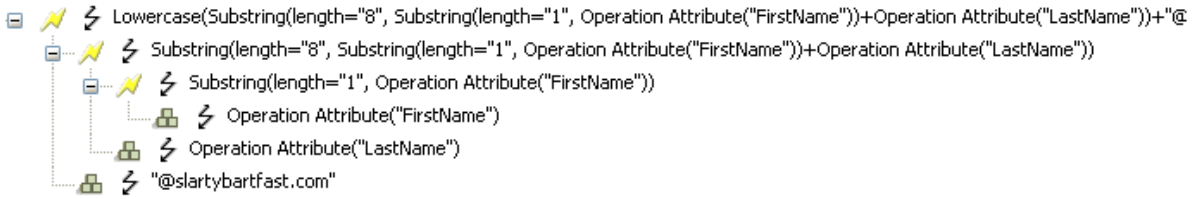
Length

Number of characters from the start to include in the substring. Negative numbers are interpreted as (total # of characters + length) + 1. For example, -1 represents the entire length of the original string. If -2 is specified, the length is the entire string -1. For a string with 5 characters, a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

Example

This example sets the e-mail address to be name@slartybartfast.com where the name equals the first character of the Given Name plus the Surname. The policy name is Policy: Create E-mail from Given Name and Surname, and it is available for download at the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `001-Command-SetEmailByGivenNameAndSurname.xml` (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).





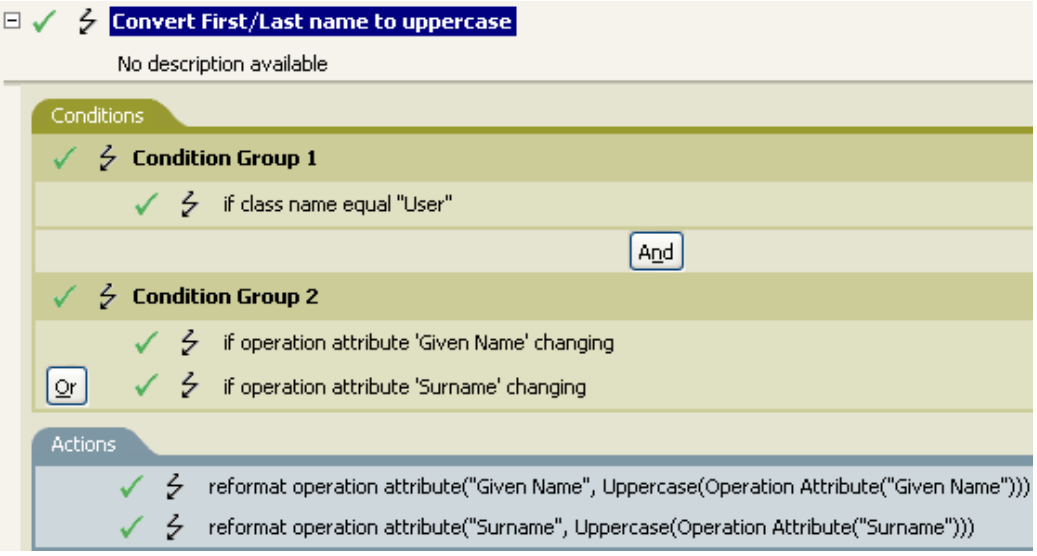
The Substring token is used twice in the action Set Destination Attribute Value. It takes the first character of the First Name attribute and adds eight characters of the Last Name attribute to form one substring.

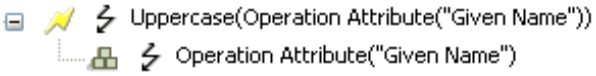
Uppercase

Converts the characters in the enclosed tokens to uppercase.

Example

The example converts the first and last name attributes of the User object to uppercase. The policy name is Policy: Convert First/Last Name to Uppercase and it is available for download at the OpenText Support Web site. For more information, see "Downloading Identity Manager Policies" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Understanding Policies Guide*. To view the policy in XML, see `002-Command-UppercaseNames.xml` (../samples/002-Command-UppercaseNames.xml).





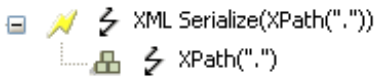
XML Parse

Parses the result of the enclosed tokens as XML and returns the resulting document node in a node set. If the result of the enclosed tokens is not well-formed XML or cannot be parsed for any reason, an empty node set is returned.

Example

```
XML Parse(Base64 Decode(charset="UTF-8", Operation Attribute("data")))
    Base64 Decode(charset="UTF-8", Operation Attribute("data"))
        Operation Attribute("data")
```

XML Serialize

Serializes the node set result of the enclosed tokens as XML. Depending on the content of the node set, the resulting string is either a well-formed XML document or a well-formed parsed general entity. If a node set is passed, it returns a string representation of the node set. If a string value of a node set is passed, it returns a string of valid tokens escaped of XML or HTML characters that can be included in a text node. For example, you can use the XML Serialize token to trace a node set and return a particular variable in a readable format, which can be useful for debugging purposes. You can also use XML Serialize to escape special characters, including `&`, `&`, `<`, or `<`.

Example