Novell
Policies in Designer 2.1

2.1

www.novell.com

POLICIES IN DESIGNER 2.1

September 28, 2007

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the Novell International Trade Services Web page (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the Novell Legal Patents Web page (http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide

Novell® Identity Manager 3.5.1 is a data sharing and synchronization service that enables applications, directories, and databases to share information. It links scattered information and enables you to establish policies that govern automatic updates to designated systems when identity changes occur.

Identity Manager provides the foundation for account provisioning, security, single sign-on, user self-service, authentication, authorization, automated workflows, and Web services. It allows you to integrate, manage, and control your distributed identity information so you can securely deliver the right resources to the right people.

This guide provides detailed reference on Policy Builder and Driver Configuration using Designer for Identity Manager 3.5.1.

- Chapter 1, "Overview," on page 17
- Chapter 3, "Managing Policies with the Policy Builder," on page 21
- Chapter 4, "Using Additional Builders," on page 47
- Chapter 5, "Using the XPath Builder," on page 65
- Chapter 6, "Defining Schema Mapping Policies," on page 73
- Chapter 7, "Controlling the Flow of Objects with the Filter," on page 87
- Chapter 8, "Using Predefined Rules," on page 101
- Chapter 9, "Testing Policies with the Policy Simulator," on page 135
- Chapter 10, "Storing Information in Resource Objects," on page 143
- Chapter 11, "Using ECMAScript in Policies," on page 153
- Chapter 12, "Conditions," on page 169
- Chapter 13, "Actions," on page 213
- Chapter 14, "Noun Tokens," on page 291
- Chapter 15, "Verb Tokens," on page 331

There is a reference section for the pre-Identity Manager 3.5 Policy Builder.

- Chapter 2, "Using the Pre-Identity Manager 3.5 Policy Builder," on page 19
- Chapter 16, "Pre-Identity Manager 3.5 Builders," on page 353
- Chapter 17, "Pre-Identity Manager 3.5 Conditions," on page 369
- Chapter 18, "Pre-Identity Manager 3.5 Actions," on page 397
- Chapter 19, "Pre-Identity Manager 3.5 Noun Tokens," on page 453
- Chapter 20, "Pre-Identity Manager 3.5 Verb Tokens," on page 479

Audience

This guide is intended for Identity Manager administrators.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of *Policies in Designer*, visit the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm35).

Additional Documentation

For documentation on using the Identity Manager drivers, see the Identity Manager Drivers Documentation Web site (http://www.novell.com/documentation/idm35drivers/index.html).

For documentation on using Designer, see the Designer 2.1 for Identity Manager 3.5.1 Documentation Web site (http://www.novell.com/documentation/designer21/).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX*, should use forward slashes as required by your software.

Overview

1

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 Policies for Identity Manager 3.5.1". 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. The following sections explain how to use the tools that are provided to help manage the policies:

- Chapter 3, "Managing Policies with the Policy Builder," on page 21
- Chapter 4, "Using Additional Builders," on page 47
- Chapter 5, "Using the XPath Builder," on page 65
- Chapter 6, "Defining Schema Mapping Policies," on page 73
- Chapter 7, "Controlling the Flow of Objects with the Filter," on page 87
- Chapter 8, "Using Predefined Rules," on page 101
- Chapter 9, "Testing Policies with the Policy Simulator," on page 135
- Chapter 10, "Storing Information in Resource Objects," on page 143
- Chapter 11, "Using ECMAScript in Policies," on page 153

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" in the *Understanding Policies for Identity Manager 3.5.1*.

- Chapter 12, "Conditions," on page 169
- Chapter 13, "Actions," on page 213
- Chapter 14, "Noun Tokens," on page 291
- Chapter 15, "Verb Tokens," on page 331

There are additional reference chapters for the pre-Identity Manager Policy Builder:

- Chapter 2, "Using the Pre-Identity Manager 3.5 Policy Builder," on page 19
- Chapter 16, "Pre-Identity Manager 3.5 Builders," on page 353
- Chapter 17, "Pre-Identity Manager 3.5 Conditions," on page 369
- Chapter 18, "Pre-Identity Manager 3.5 Actions," on page 397
- Chapter 19, "Pre-Identity Manager 3.5 Noun Tokens," on page 453
- Chapter 20, "Pre-Identity Manager 3.5 Verb Tokens," on page 479

1.1 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 12, "Conditions," on page 169) that must be met before a defined action (see Chapter 13, "Actions," on page 213) occurs.

- Actions can have dynamic arguments that derive from tokens that are expanded at run time.
- Tokens are broken up into two classifications: nouns and verbs.
  - Noun tokens (see Chapter 14, “Noun Tokens,” on page 291) expand to values that are derived from the current operation, the source or destination data stores, or some external source.
  - Verb tokens (see Chapter 15, “Verb Tokens,” on page 331) modify the concatenated results of other tokens that are subordinate to them.
- Regular expressions (see “Regular Expressions” in *Understanding Policies for Identity Manager 3.5.1*) and XPath 1.0 expressions (see “XPath 1.0 Expressions” in the *Understanding Policies for Identity Manager 3.5.1*) 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 the Novell® `nds.dtd`; for more information, see “NDS DTD” in the *Identity Manager 3.5.1 DTD Reference*.
- 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.

Using the Pre-Identity Manager 3.5 Policy Builder 2

Designer contains two policy builder, the pre-Identity Manager 3.5 Policy Builder and the Identity Manager 3.5 and later Policy Builder. The Policy Builders are similar except for:

- You can only enable and disable trace at the driver level in the pre-Identity Manager 3.5 Policy Builder.
- The DirXML Script elements are different between the two builders.

These differences require two Policy Builders. For information on how to use both Policy Builders, see Chapter 3, “Managing Policies with the Policy Builder,” on page 21. This documents the Identity Manager 3.5 and Later Policy Builder. The only difference is an additional icon that enables and disables tracing on rules, actions, conditions, and tokens.

For a list of the DirXML Script elements for the pre-Identity Manager 3.5 Policy Builder:

- Chapter 17, “Pre-Identity Manager 3.5 Conditions,” on page 369
- Chapter 18, “Pre-Identity Manager 3.5 Actions,” on page 397
- Chapter 19, “Pre-Identity Manager 3.5 Noun Tokens,” on page 453
- Chapter 20, “Pre-Identity Manager 3.5 Verb Tokens,” on page 479

For a list of the DirXML Script elements for the Identity Manager 3.5 and later Policy Builder:

- Chapter 12, “Conditions,” on page 169
- Chapter 13, “Actions,” on page 213
- Chapter 14, “Noun Tokens,” on page 291
- Chapter 15, “Verb Tokens,” on page 331

Managing Policies with the Policy Builder

3

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

- Section 3.1, "Accessing the Policy Builder," on page 21
- Section 3.2, "Using the Policy Builder," on page 25
- Section 3.3, "Creating a Policy," on page 26
- Section 3.4, "Creating a Rule," on page 32
- Section 3.5, "Creating an Argument," on page 40
- Section 3.6, "Editing a Policy," on page 41
- Section 3.7, "Viewing the Policy in XML," on page 45

3.1 Accessing the Policy Builder

Designer contains two policy builder, the pre-Identity Manager 3.5 Policy Builder and the Identity Manager 3.5 and Later Policy Builder. The Policy Builder version is determined by the version of Identity Manager. To set the version of Identity Manager:

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

   The options are:

   - 2.0
   - 2.0.1
   - 2.0.2
   - 3.0
   - 3.0.1
   - 3.5
   - 3.5.1

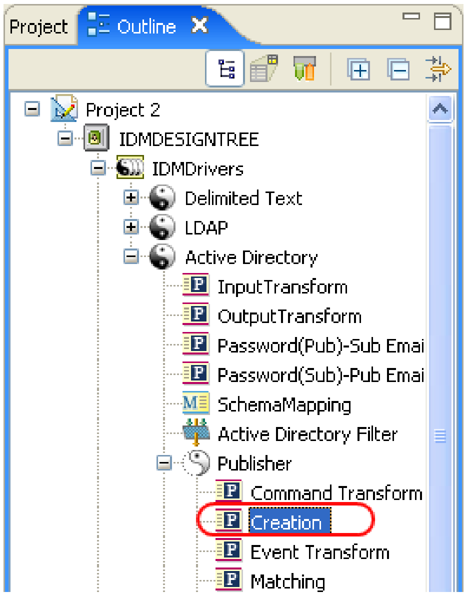
When the Identity Manager version is set to 3.5 or above, the new Policy Builder is available. If the version is set to anything below 3.5, the old Policy Builder is available.

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

- Section 3.1.1, "Model Outline View," on page 22
- Section 3.1.2, "Policy Flow View," on page 22
- Section 3.1.3, "Policy Set," on page 23

3.1.1 Model Outline View

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

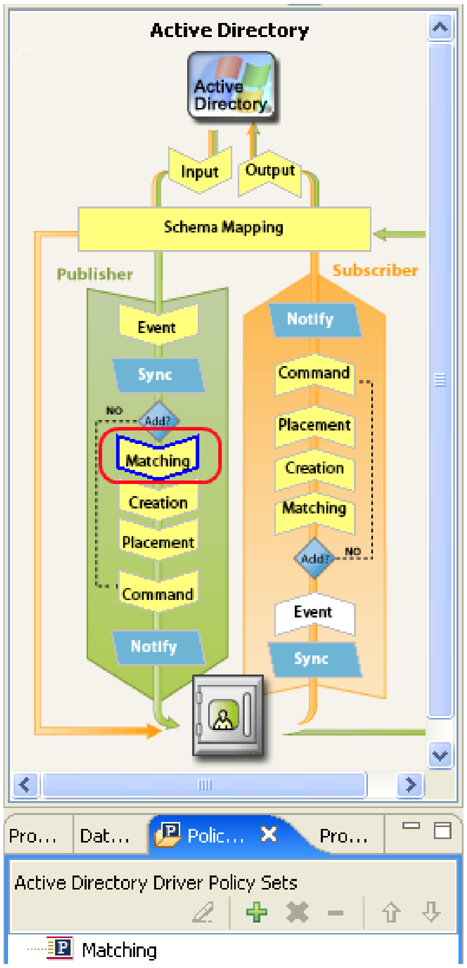


3.1.2 Policy Flow View

1. Open a project in Designer.
2. Select the *Outline* tab > select the *Show Policy Flow* icon.
3. Right-click a policy (for example, the Matching policy) in the Policy Flow view, then select *Edit Policy*.

   or

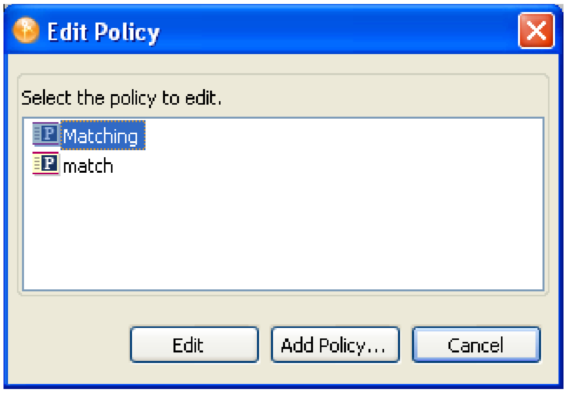
Double-click the Matching policy in the Policy Flow



Active Directory Driver Policy Sets

Matching

4 Select the policy, then click *Edit.*



Select the policy to edit.

Matching

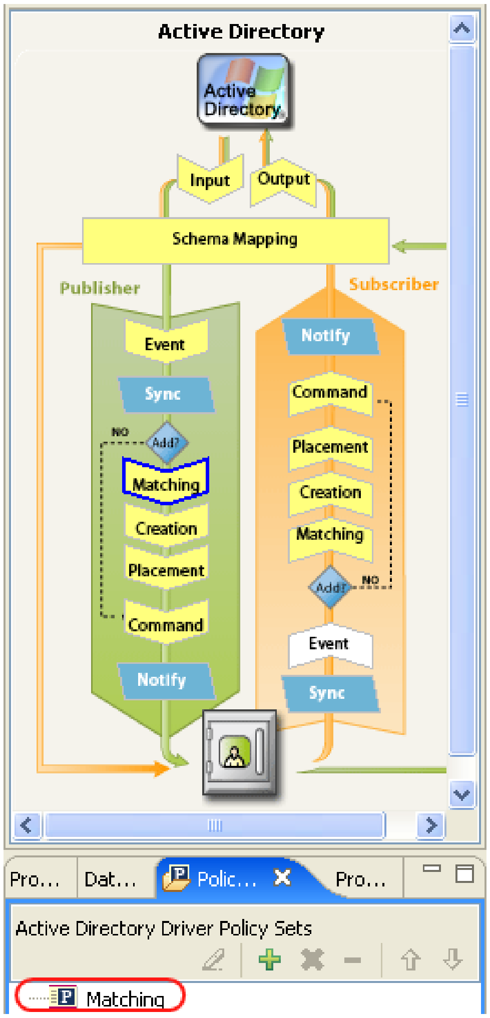
match

3.1.3 Policy Set

1 Right-click the policy in the policy set, then click *Edit*.

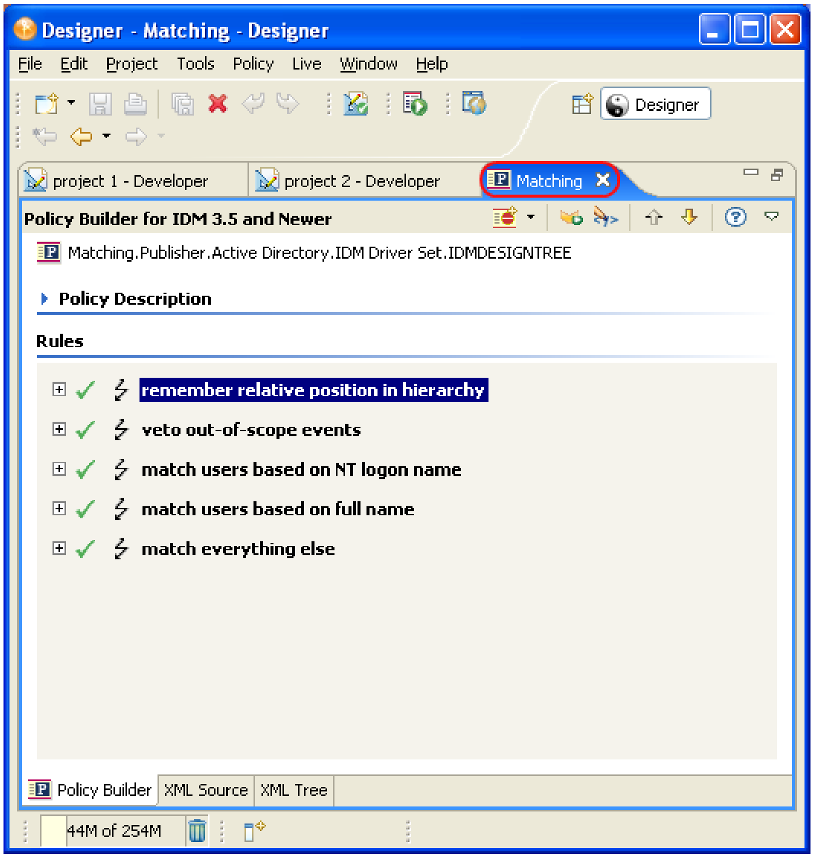
or

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



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 3-1 Policy Builder Full Screen


Designer - Matching - Designer
File Edit Project Tools Policy Live Window Help
Designer
project 1 - Developer
project 2 - Developer
Matching
Policy Builder for IDM 3.5 and Newer
Matching.Publisher.Active Directory.IDM Driver Set.IDMDESIGNTREE
Policy Description
Rules
Policy Builder
XML Source
XML Tree
44M of 254M

- remember relative position in hierarchy
- veto out-of-scope events
- match users based on NT logon name
- match users based on full name
- match everything else

For information on using the Policy Builder, see Section 3.2, "Using the Policy Builder," on page 25.

3.2 Using the Policy Builder

The Policy Builder enables you to add, view, and delete the rules that make up a policy. You can also import and save policies and rules, and manage XML namespaces using the Policy Builder. The Policy Builder contains the "Action Builder" on page 47 and the "Condition Builder" on page 53.

3.2.1 Tips

- Click ✓ icon to disable a policy, rule, condition, or action. Click ⊖ icon to re-enable it.
- Click the *Disable Trace* icon to disable tracing on a policy, rule, actions, conditions, or token. Click the *Enable Trace* icon to re-enable tracing.
- Click the *Edit* icon ✎ to edit the name of a rule or edit the description of a rule.
- Click the 🔍 icon to see a list of values for a field.
- Click the *Delete* icon ✗ to delete a rule or a policy.
- Click the *Rule* icon to add a new rule. If you click the down-arrow on the right, you can add a rule, use predefined rules, include a policy, or create a new Condition Group.

- Click the *Import Policy from file* icon to import a policy or a rule. Click the *Save to File* icon to export a policy or a rule.
- Click the *Edit Namespace* icon to add multiple XML namespaces to the rule or policy.
- Click the *Launch XPath Builder* icon to launch the XPath Builder. It allows you to create XPath expressions.
- Click ⊞ to expand all of the rules or click ⊟ to collapse all of the rules.
- Click ⇧ icon to move a rule up. Click ⇩ icon to move the rule down.
- To save your work from the main menu, click *File* > *Save* or press Ctrl+S.
- To add a comment to a policy or rule, fill in the *Policy Description* field. Comments are stored directly in the policy or rule, and can be as long as necessary.

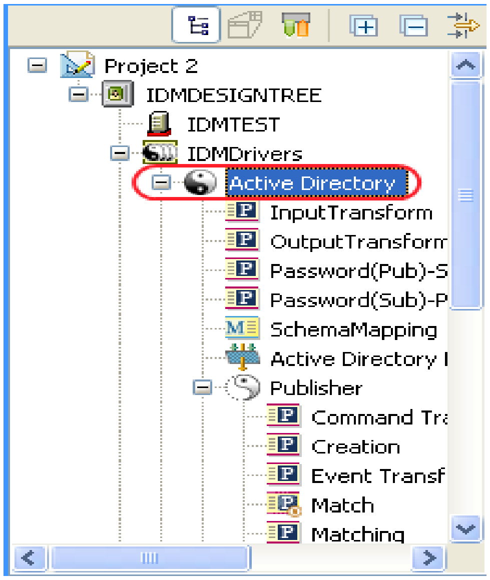
3.3 Creating a Policy

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

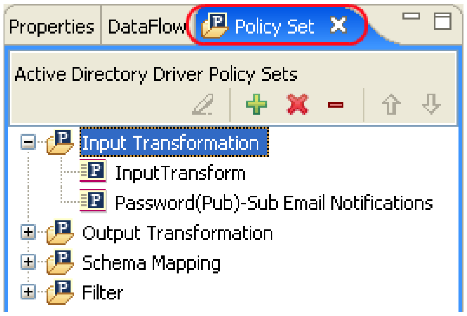
- Section 3.3.1, "Accessing the Policy Set," on page 26
- Section 3.3.2, "Using the Policy Set," on page 27
- Section 3.3.3, "Using the Add Policy Wizard," on page 28

3.3.1 Accessing the Policy Set

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



2 Select the *Policy Set* tab.

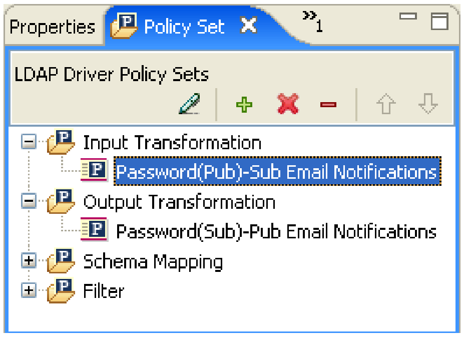


3.3.2 Using the Policy Set

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.

Figure 3-2 *Policy Set*



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 3-1 *Policy Set Toolbar*

Operation	Description
Edit a 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, do not delete ▬	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.

Operation	Description
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 3-2 *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.

3.3.3 Using the Add Policy Wizard

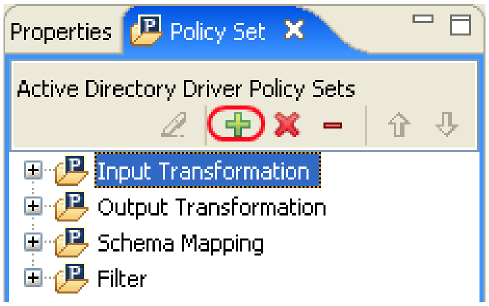
The Add Policy Wizard launches when you click the *Create or add a new policy to the Policy Set* icon in the toolbar. The Add Policy Wizard enables you to do the following:

- "Creating a Policy" on page 29
- "Copying a Policy" on page 31
- "Linking to a Policy" on page 31

To launch the Add Policy Wizard:

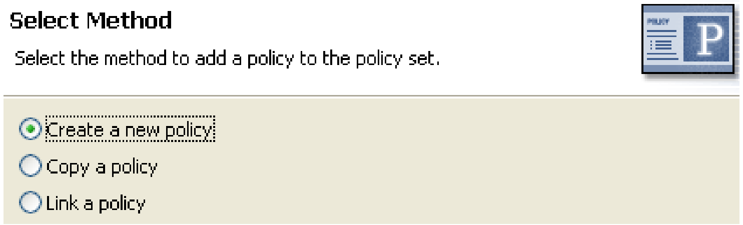
1 Select a driver in the *Outline* view.

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



Creating a Policy

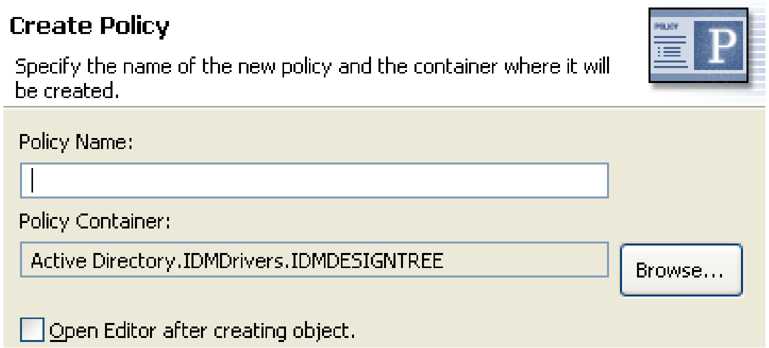
1 In the Add Policy Wizard, select *Create a new policy*, then click *Next*.


Select Method

Select the method to add a policy to the policy set.

● Create a new policy
○ Copy a policy
○ Link a policy

2 Provide a policy name.


Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Policy Container: Active Directory.IDMDrivers.IDMDESIGNTREE

□ Open Editor after creating object.

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

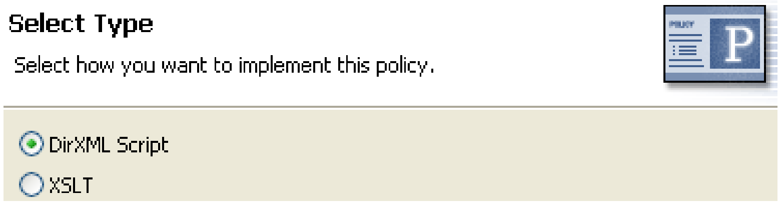
If a policy is not reused by multiple drivers, you typically create that policy under the driver or channel that is using it.

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.

4. Select the type of policy you want to implement. The policy type defaults to *DirXML Script*. You can select *XSLT*, if you don't want to use DirXML® Script.



Select Type

Select how you want to implement this policy.

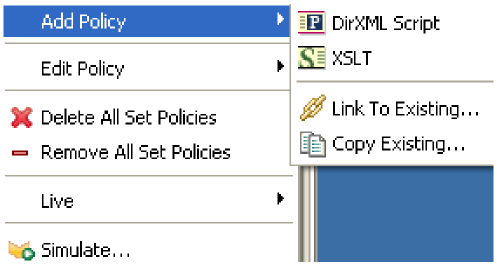
● DirXML Script
○ XSLT

5. Click *Finish*.

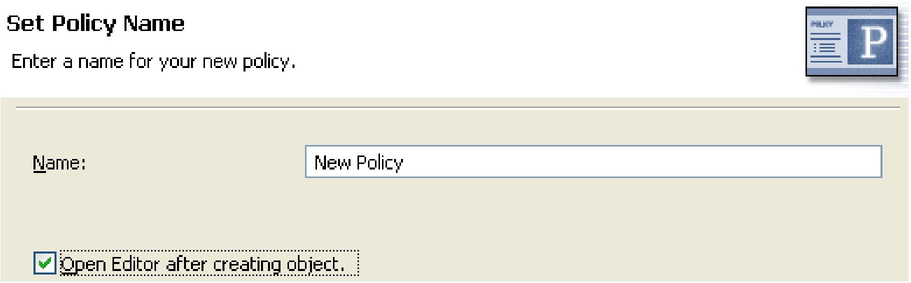
If you create a Schema Mapping policy set, then an additional option is available for *Schema Mapping*. The new policy appears in the expanded policy set.

You can also add a policy by right-clicking a policy set in the Policy Flow view.

1. Right-click a policy set (for example, Input Transformation Set).
2. Select *Add Policy*.
3. Select how to implement the policy.
   - *DirXML Script*
   - *XSLT*
   - *Link To Existing*
   - *Copy Existing*
   - *Schema Mapping* (Only displayed, if the Schema Mapping policy set is selected.)



4. Name the policy.



Set Policy Name

Enter a name for your new policy.

Name: New Policy

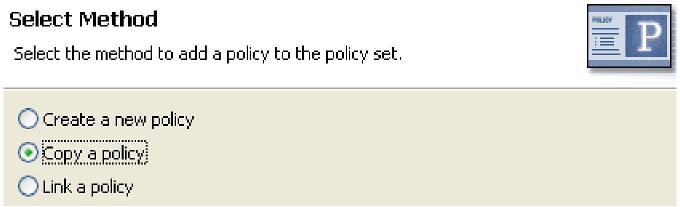
☑ Open Editor after creating object.

5. Click *Open Editor after creating policy*.

6 Click *OK*.

Copying a Policy

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



Select Method

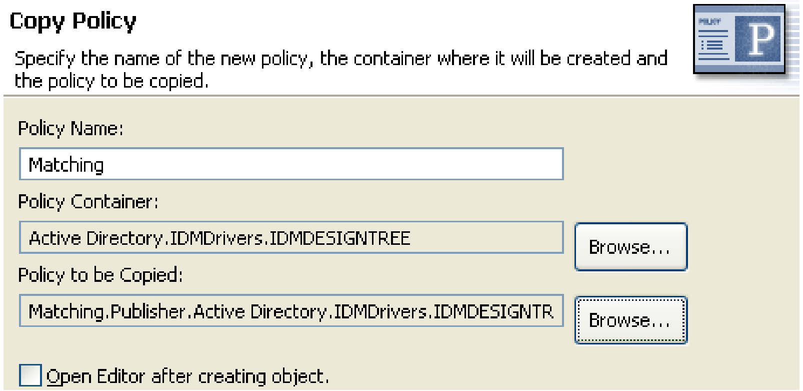
Select the method to add a policy to the policy set.

○ Create a new policy
● Copy a policy
○ Link a policy

2 Name the policy.

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

4 Browse to and select the policy you want to copy, then click *OK*.



Copy Policy

Specify the name of the new policy, the container where it will be created and the policy to be copied.

Policy Name: Matching

Policy Container: Active Directory.IDMDrivers.IDMDESIGNTREE

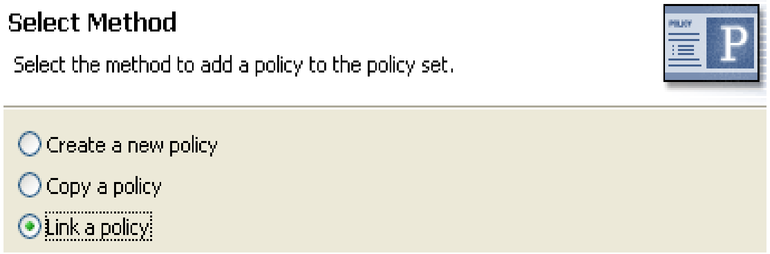
Policy to be Copied: Matching.Publisher.Active Directory.IDMDrivers.IDMDESIGNTR

□ Open Editor after creating object.

5 Click *Finish* to make a copy of the selected policy.

Linking to a Policy

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

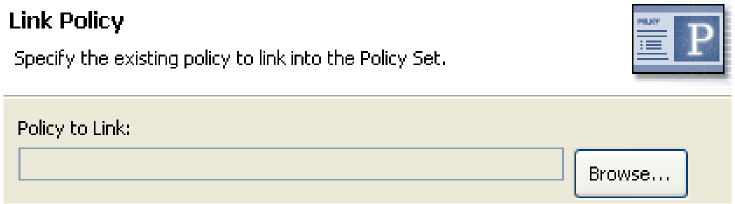


Select Method

Select the method to add a policy to the policy set.

○ Create a new policy
○ Copy a policy
● Link a policy

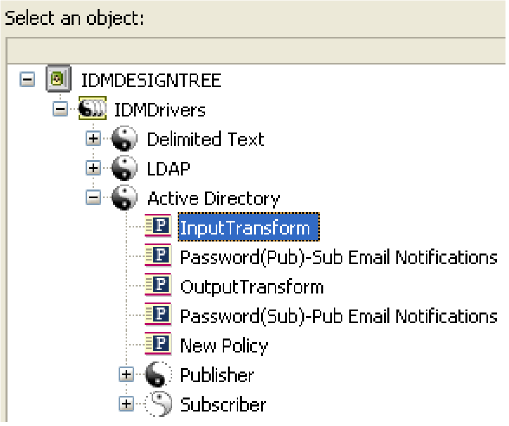
2 Click *Browse* to launch the model browser.



Link Policy

Specify the existing policy to link into the Policy Set.

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



Select an object:

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 Mapping 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.

3.4 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:

- Section 3.4.1, "Creating a New Rule," on page 33
- Section 3.4.2, "Using Predefined Rules," on page 37
- Section 3.4.3, "Including an Existing Rule," on page 38
- Section 3.4.4, "Importing a Policy From an XML File," on page 39

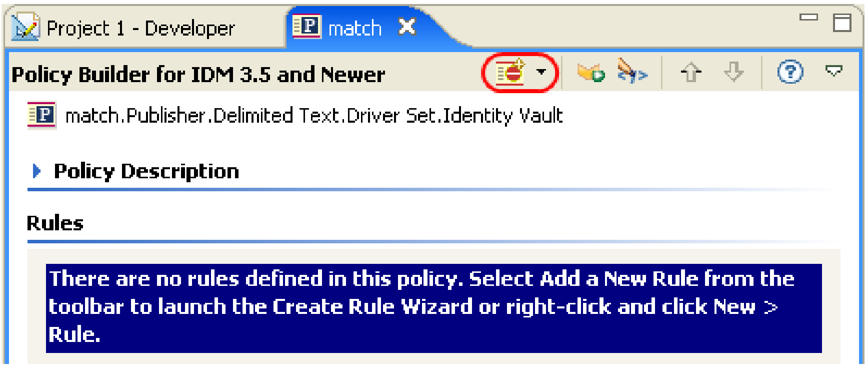
3.4.1 Creating a New Rule

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 33
- "Creating a Conditional Group" on page 36
- "Creating a Condition" on page 37
- "Creating an Action" on page 37

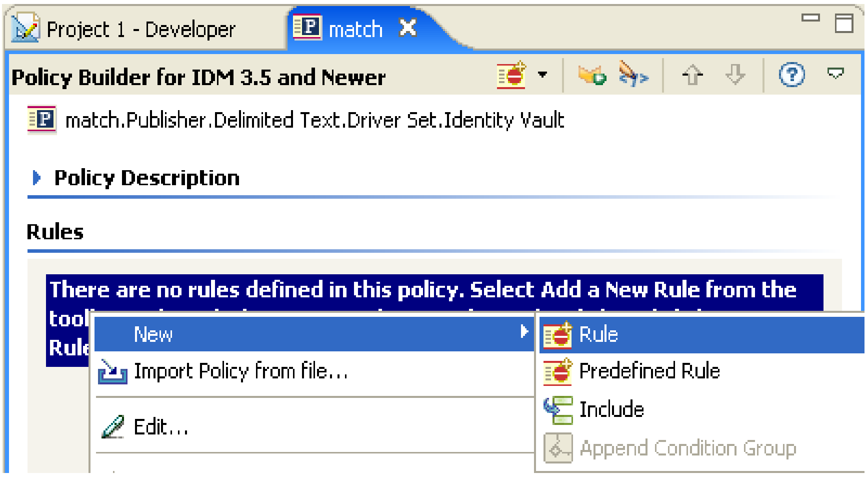
Creating a Rule

1 From the Policy Builder toolbar, select *Rule*.


Project 1 - Developer
match
Policy Builder for IDM 3.5 and Newer
match.Publisher.Delimited Text.Driver Set.Identity Vault
Policy Description
Rules

There are no rules defined in this policy. Select Add a New Rule from the toolbar to launch the Create Rule Wizard or right-click and click New > Rule.

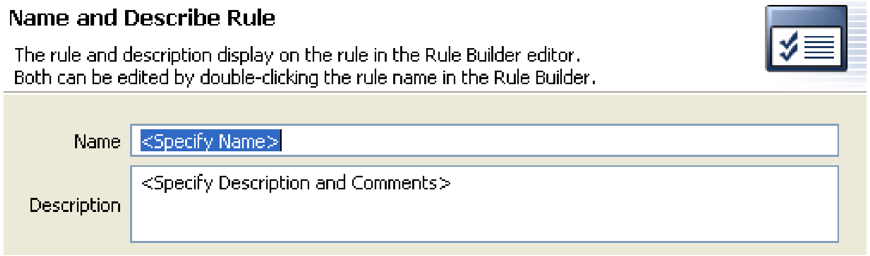
You can also right-click and click *New* > *Rule*.


Project 1 - Developer
match
Policy Builder for IDM 3.5 and Newer
match.Publisher.Delimited Text.Driver Set.Identity Vault
Policy Description
Rules
New
Import Policy from file...
Edit...
Rule
Predefined Rule
Include
Append Condition Group

There are no rules defined in this policy. Select Add a New Rule from the tool... Rul...

Either option launches the Create Rule Wizard.

2 Specify the name of the rule, then click *Next*.



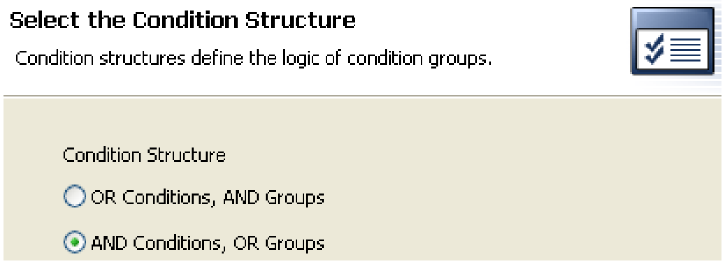
Name and Describe Rule

The rule and description display on the rule in the Rule Builder editor. Both can be edited by double-clicking the rule name in the Rule Builder.

Name: <Specify Name>

Description: <Specify Description and Comments>

3 Select the condition structure (*OR Conditions, AND Groups* or *AND Conditions, OR Groups*) then click *Next*.



Select the Condition Structure

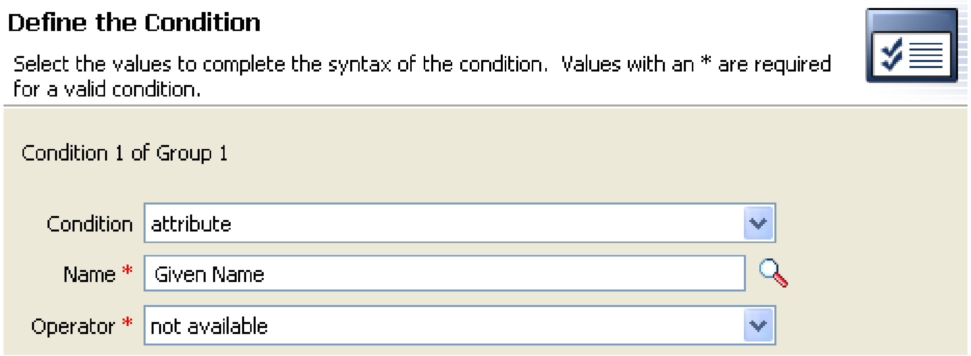
Condition structures define the logic of condition groups.

Condition Structure

○ OR Conditions, AND Groups

● AND Conditions, OR Groups

4 Select the condition you want, specify the appropriate information, then click *Next*.



Define the Condition

Select the values to complete the syntax of the condition. Values with an * are required for a valid condition.

Condition 1 of Group 1

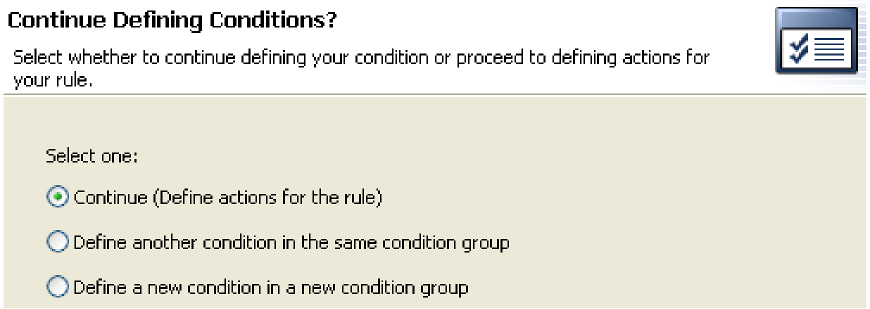
Condition: attribute

Name *: Given Name

Operator *: not available

Click the Help icon ⓘ for information about each condition you can create.

5 You can define an additional condition or condition group at this point. For this example, there is only one condition. Select *Continue*, then click *Next*.



Continue Defining Conditions?

Select whether to continue defining your condition or proceed to defining actions for your rule.

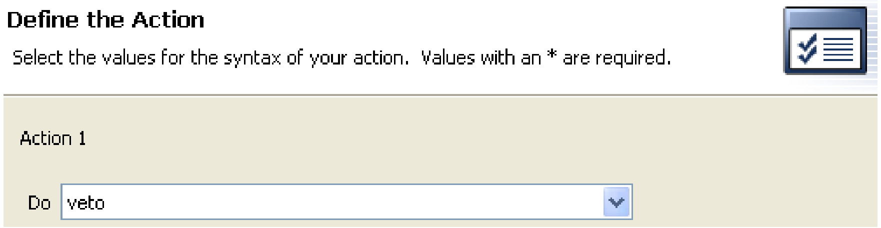
Select one:

● Continue (Define actions for the rule)

○ Define another condition in the same condition group

○ Define a new condition in a new condition group

6 Select the action that you want, then click *Next*.



Define the Action

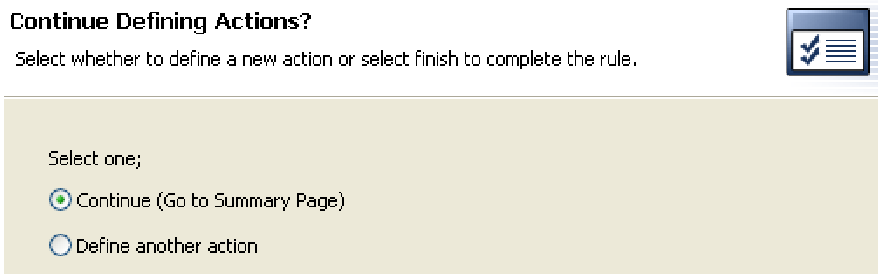
Select the values for the syntax of your action. Values with an * are required.

Action 1

Do veto

Click the Help icon ⓘ for information about each action you can create.

7 You can define additional actions at this point. For this example, there is only one action. Select *Continue*, then click *Next*.



Continue Defining Actions?

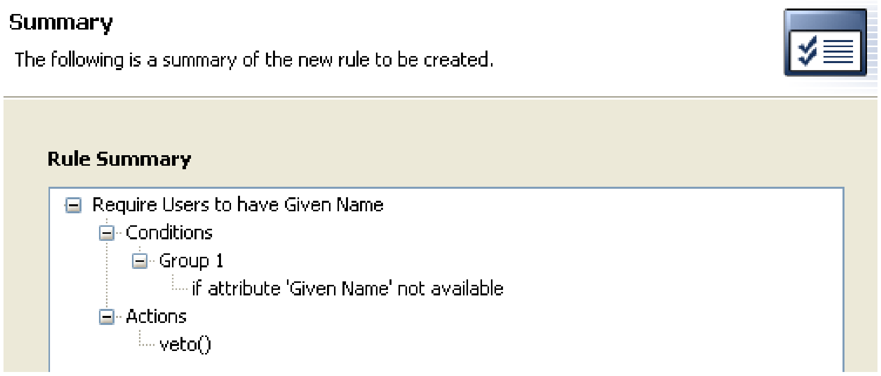
Select whether to define a new action or select finish to complete the rule.

Select one;

● Continue (Go to Summary Page)

○ Define another action

8 The summary page displays the rule that was created. Click *Finish* to complete the creation of the rule.



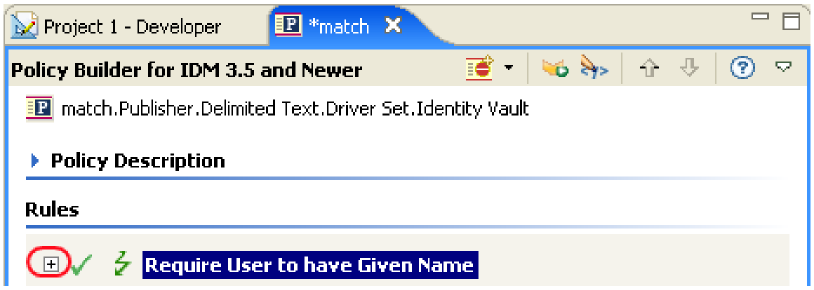
Summary

The following is a summary of the new rule to be created.

Rule Summary

```
⊟ Require Users to have Given Name
    ⊟ Conditions
        ⊟ Group 1
            if attribute 'Given Name' not available
    ⊟ Actions
        veto()
```

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



Policy Builder for IDM 3.5 and Newer

match.Publisher.Delimited Text.Driver Set.Identity Vault

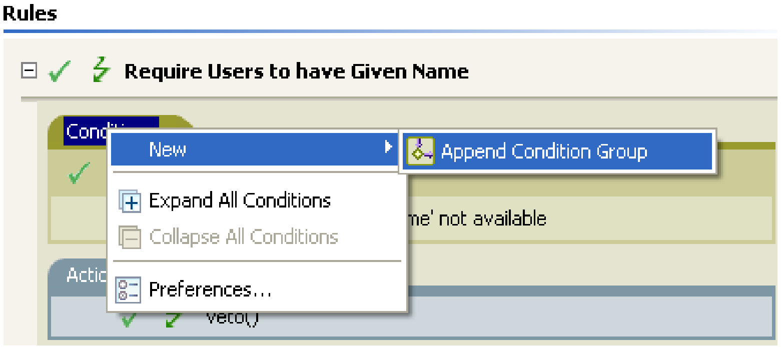
Policy Description

Rules

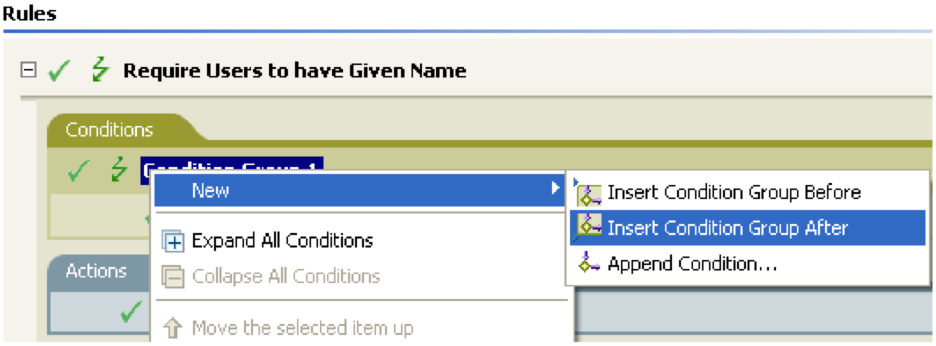
⊞ ✓ **Require User to have Given Name**

Creating a Conditional Group

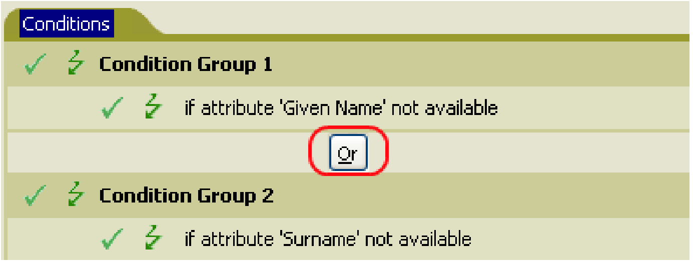
1 Right-click the *Conditions* tab or right-click the name of the *Condition Group*, then click *New > Insert Condition Group Before* or *Insert Condition Group After*.



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

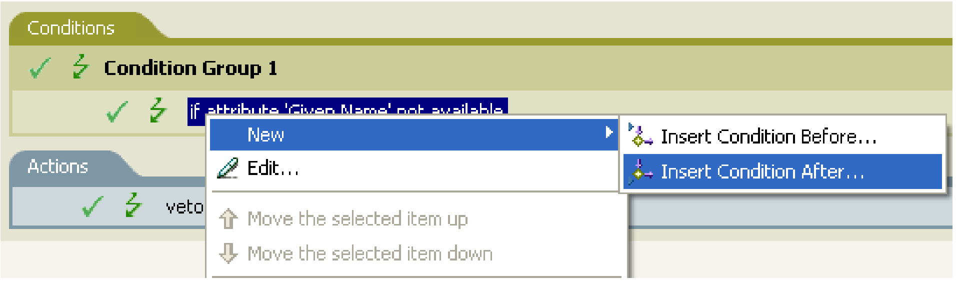

Rules
Require Users to have Given Name
Conditions
Condition Group 1
New
Insert Condition Group Before
Insert Condition Group After
Append Condition...
Expand All Conditions
Collapse All Conditions
Actions
Move the selected item up

You can change the condition for the Condition Groups by click the *And/Or* icon.

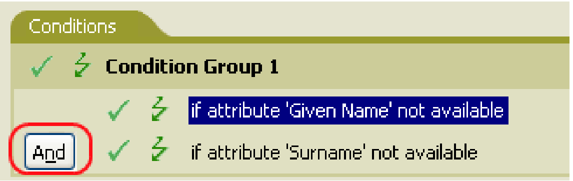

Conditions
Condition Group 1
if attribute 'Given Name' not available
Or
Condition Group 2
if attribute 'Surname' not available

Creating a Condition

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

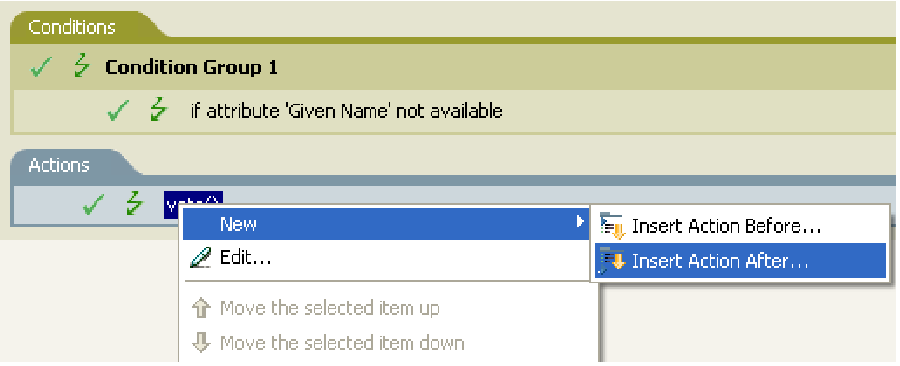

Conditions
Condition Group 1
if attribute 'Given Name' not available
Actions
veto
New
Edit...
Move the selected item up
Move the selected item down
Insert Condition Before...
Insert Condition After...

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


Conditions
Condition Group 1
if attribute 'Given Name' not available
And
if attribute 'Surname' not available

Creating an Action

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

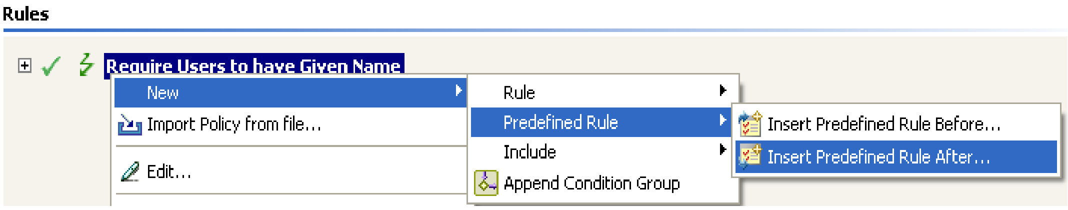


3.4.2 Using Predefined Rules

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* > *Predefine Rules* > *Insert Predefined Rule Before* or *Insert Predefined Rule After*.

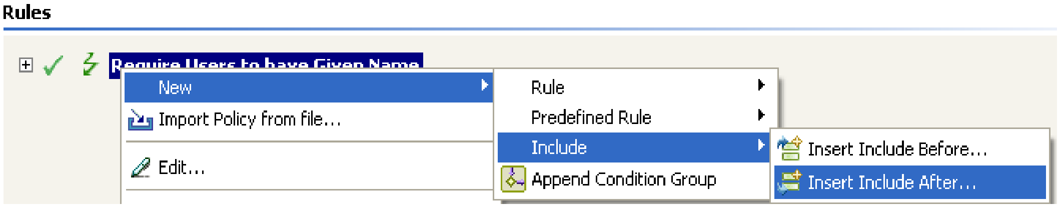
See Chapter 8, “Using Predefined Rules,” on page 101 for more information.


Rules
Require Users to have Given Name
New
Import Policy from file...
Edit...
Rule
Predefined Rule
Include
Append Condition Group
Insert Predefined Rule Before...
Insert Predefined Rule After...

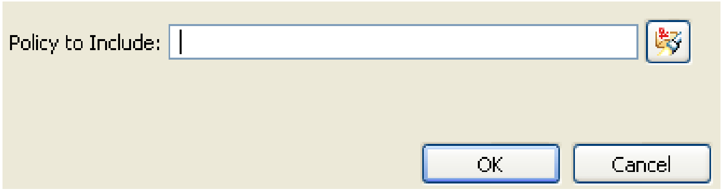
3.4.3 Including an Existing Rule

Designer allows you to include the rules from another policy.

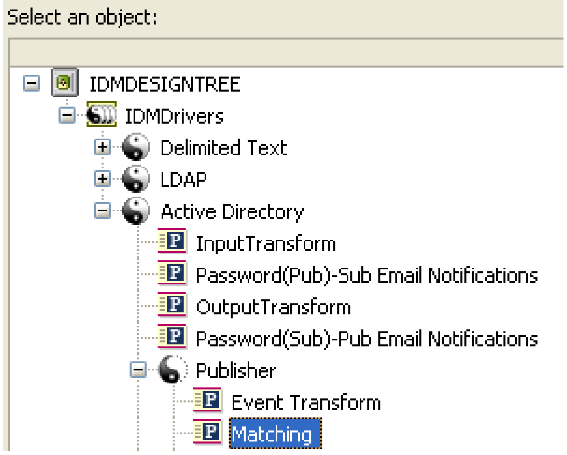
1 Right-click in the Policy Builder and click *New* > *Include* > *Insert Include Before* or *Insert Include After*.



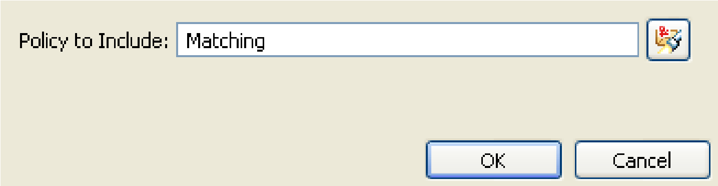
2 Click the Browse icon.



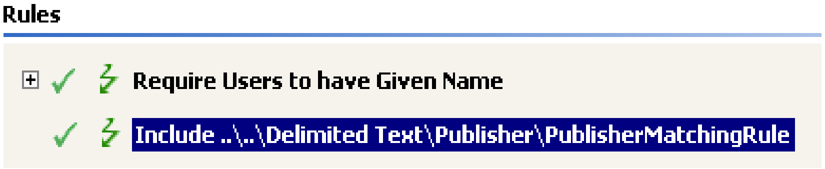
3 Browse to the policy you want to include, then click *OK*.



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


Policy to Include: Matching

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


Rules

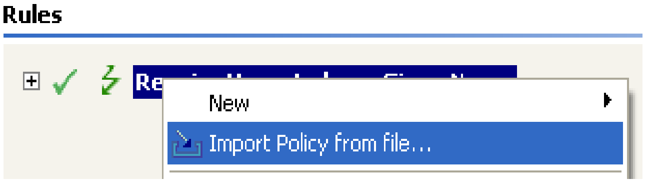
⊞ ✓ Require Users to have Given Name

✓ Include ..\..\Delimited Text\Publisher\PublisherMatchingRule

3.4.4 Importing a Policy From an XML File

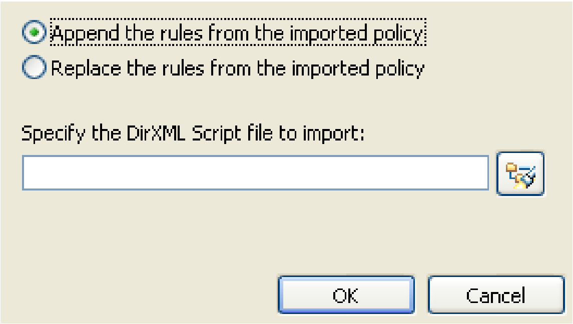
Rules and policies can be saved as XML files. If you have a file that contains a rule or a policy you want to use, the Policy Builder allows you to import the file.

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


Rules

⊞ ✓ Re

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


● Append the rules from the imported policy

○ Replace the rules from the imported policy

Specify the DirXML Script file to import:

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

4 Click *OK*.

3.5 Creating an Argument

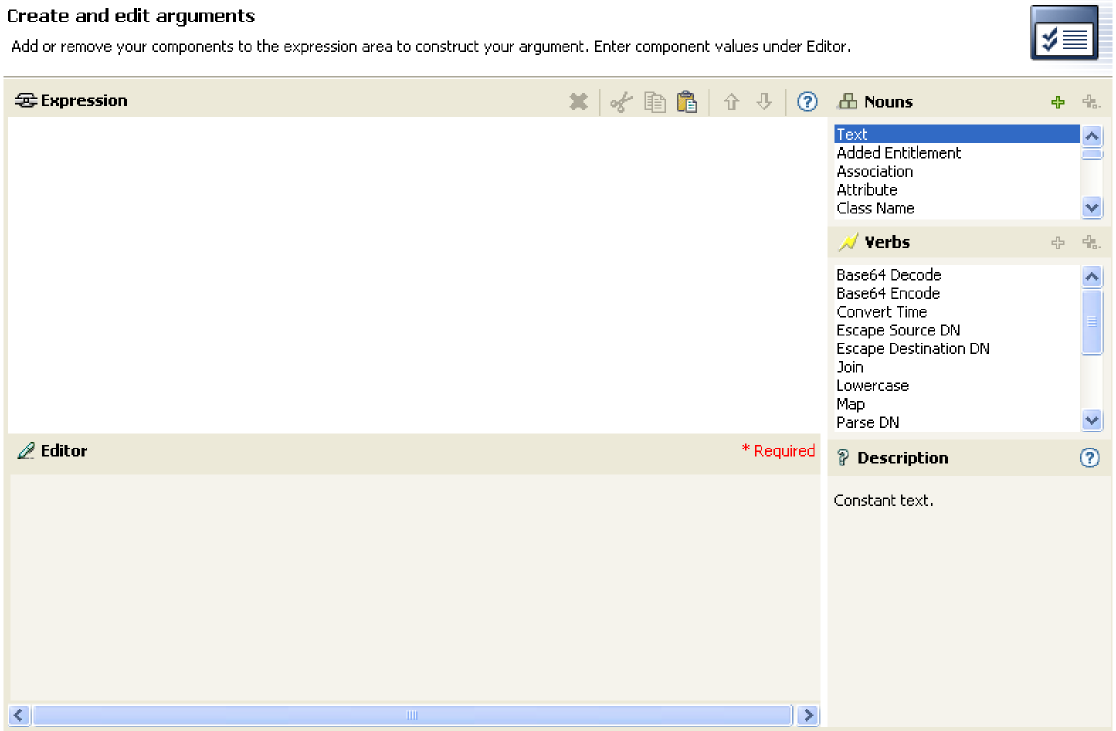
The Argument Builder provides a dynamic graphical interface that enables you to construct complex argument expressions for use within the Policy Builder. To access the Argument Builder, see "Argument Builder" on page 49.

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 nouns tokens (values, objects, variables, etc.), and combine then with verb tokens (substring, escape, uppercase, and lowercase) to construct arguments. Multiple tokens are combined to construct complex arguments.

***Figure 3-3** Argument Builder*



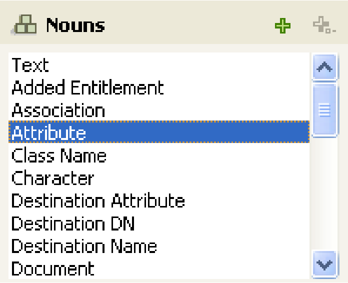
Create and edit arguments

Add or remove your components to the expression area to construct your argument. Enter component values under Editor.

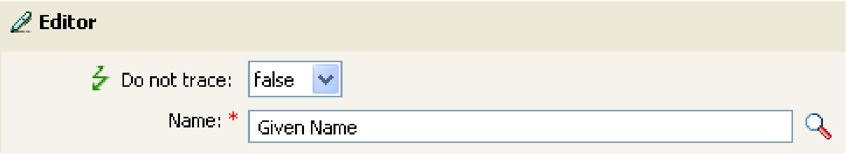
Constant text.

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* from the list of noun tokens to select it.



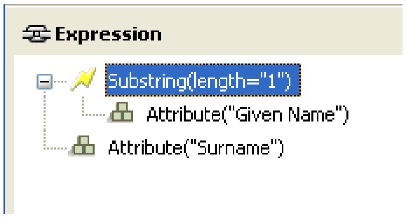
2 Browse to and select the attribute name in the *Editor* field.



Do not trace: false

Name: * Given Name

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.



```
Substring(length="1")
    Attribute("Given Name")
Attribute("Surname")
```

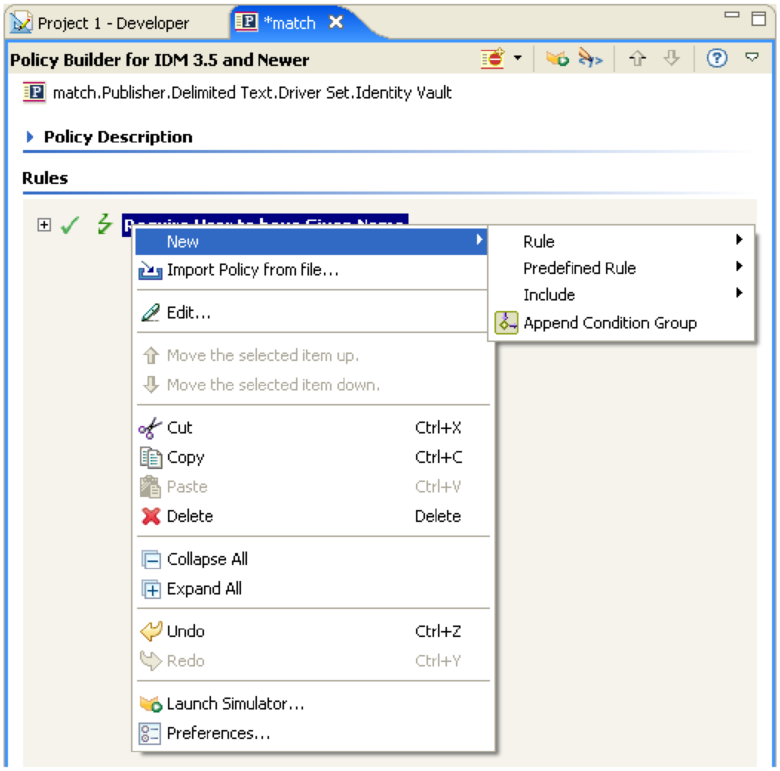
After you add a noun or verb, you can provide values in the editor, then immediately add another noun or verb. You do not need to refresh the Expression pane to apply your changes; they appear when the next operation is performed.

See "Noun Tokens" on page 291 and "Verb Tokens" on page 331 for a detailed reference on the noun and verb tokens. See "Argument Builder" on page 49 for more information on the Argument Builder.

3.6 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.

Figure 3-4 Policy Builder Context Menu and Toolbar



3.6.1 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 3-3 Policy Builder Actions and Menu Items

Operation	Description
Collapse All	Collapses all expanded rules.
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.
Disable	Displays a rule, condition, or action as disabled.
Disable Trace	Disables trace on the rule.
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*.

Operation	Description
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 Simulator	Launches the Policy Simulator.
Move and drop	Enables you to select and move an item. Select the item, then drag it.
Move the selected item down	Moves the item down in the list of policies.
Move the selected item 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.
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.

3.6.2 Keyboard Support

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

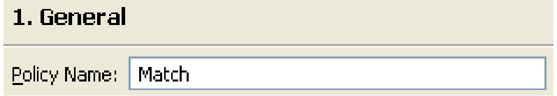
Table 3-4 *Keyboard Support in the Policy Builder*

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.

Keystroke	Description
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

3.6.3 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.



1. General

Policy Name: Match

4. Click *OK*.

3.6.4 Saving Your Work

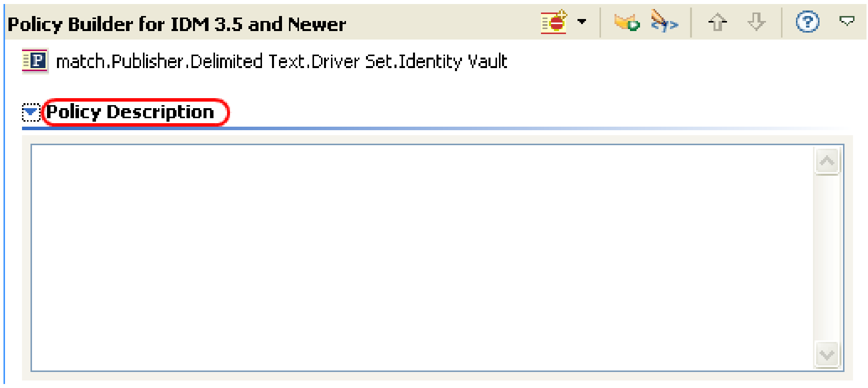
Do one of the following:

- 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.

3.6.5 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, double click *Policy Description*.



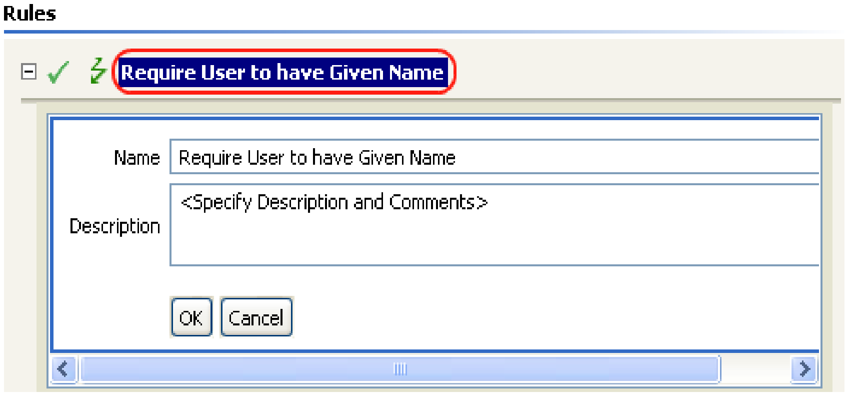
Policy Builder for IDM 3.5 and Newer
match.Publisher.Delimited Text.Driver Set.Identity Vault
Policy Description

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.



Name: Require User to have Given Name

Description: <Specify Description and Comments>

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

3 Save the rule by pressing Ctrl+S.

3.7 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 Novell XML Editor" in the *Designer 2.1 for Identity Manager 3.5.1*.

Using Additional Builders

4

Although you define most arguments using the Argument Builder, there are several more builders 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:

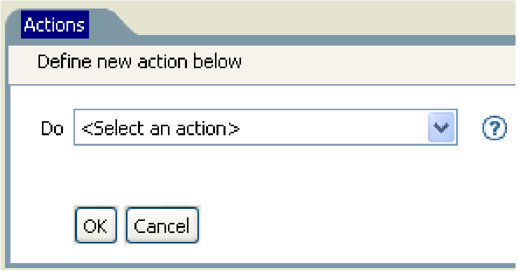
4.1 Action Builder

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

4.1.1 Creating an Action

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

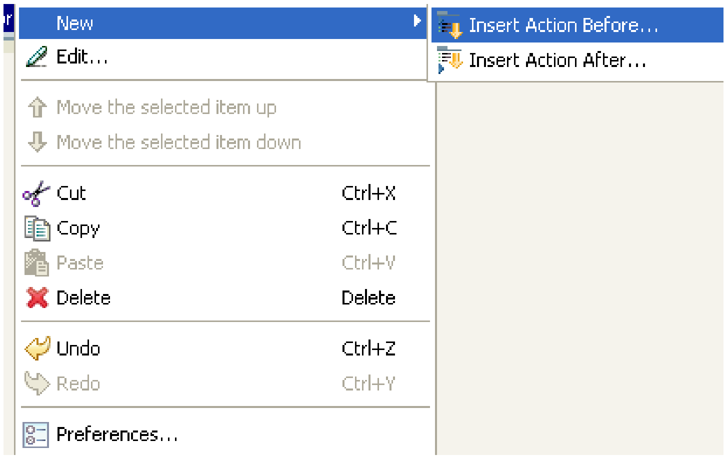
2 Double-click the *Actions* tab to launch the Action Builder.



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

4.1.2 Additional Options for the Action Builder

1 Right-click the action to see the additional options:



- **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 the selected item up:** Moves the selected action up in the order of execution.
- **Move the selected item down:** Moves the selected action down in the order of execution.
- **Cut, Copy, Paste, or Delete an Action:** Cuts, copies, pastes, or deletes the action.
- **Undo or Redo:** Undoes or redoes the last action.
- **Preferences:** Allows you to set default functionality in the Policy Builder.
- **Help:** Select an action, then click the *Help* icon to see information specific to that action.

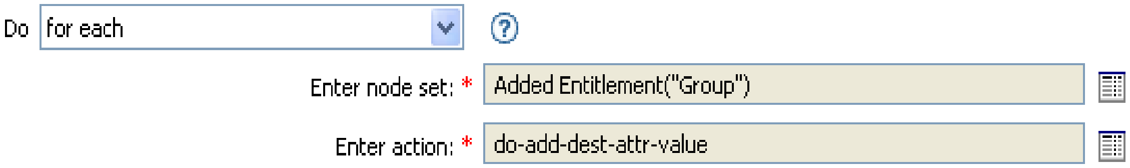
4.2 Actions Builder

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 arguments* icon ▤.

- For Each (page 239)
- Implement Entitlement (page 245)
- If (page 243)
- While (page 289)

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 4-1 *For Each Action*



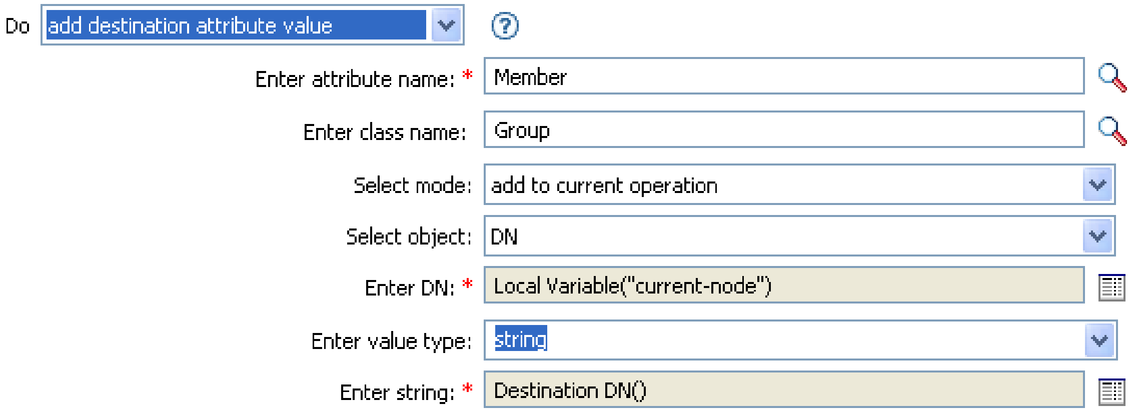
Do for each

Enter node set: * Added Entitlement("Group")

Enter action: * do-add-dest-attr-value

To define the action of the add destination attribute value, click the icon that 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 4-2** Actions Builder*



Do: add destination attribute value

Enter attribute name: * Member

Enter class name: Group

Select mode: add to current operation

Select object: DN

Enter DN: * Local Variable("current-node")

Enter value type: string

Enter string: * Destination DN()

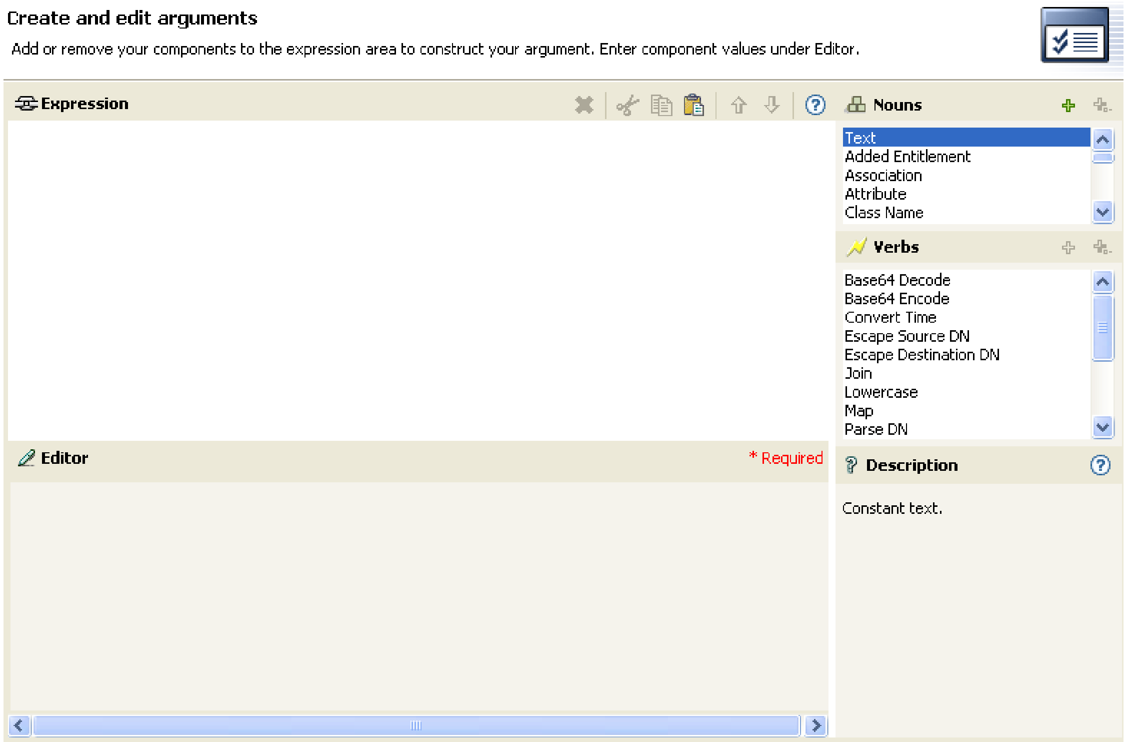
4.3 Argument Builder

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

The Argument Builder consists of five separate sections:

- **Nouns:** Contains a list of all of the available noun tokens. Select a noun token, then click *Add* to add the noun token to the *Expression* pane. See "Noun Tokens" on page 291 for more information.
- **Verbs:** Contains a list of all of the available verb tokens. Select a verb token, then click *Add* to add the verb token to the *Expression* pane. See "Verb Tokens" on page 331 for more information.
- **Description:** Contains a brief description of the 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.

Figure 4-3 *Argument Builder*



Create and edit arguments

Add or remove your components to the expression area to construct your argument. Enter component values under Editor.

Constant text.

- Section 4.3.1, "Launching the Argument Builder," on page 50
- Section 4.3.2, "Argument Builder Example," on page 51

4.3.1 Launching the Argument Builder

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

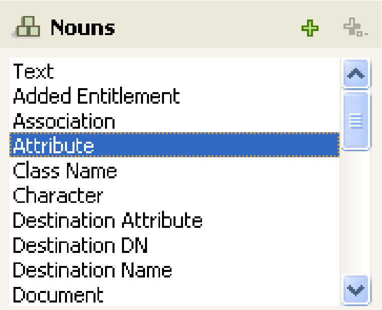
- Add Association (page 215)
- Add Destination Attribute Value (page 216)
- Add Destination Object (page 218)
- Add Source Attribute Value (page 220)
- Append XML Text (page 224)
- Clear Destination Attribute Value (page 227) (when the selected object is DN or Association)
- Clear Source Attribute Value (page 229) (when the selected object is DN or Association)
- Delete Destination Object (page 234) (when the selected object is DN or Association)
- Delete Source Object (page 235) (when the selected object is DN or Association)
- Find Matching Object (page 236)
- For Each (page 239)
- Move Destination Object (page 246)
- Move Source Object (page 248)

- Reformat Operation Attribute Value (page 249)
- Remove Association (page 251)
- Remove Destination Attribute Value (page 252)
- Remove Source Attribute Value (page 253)
- Rename Destination Object (page 254) (when the selected object is DN or Association and Enter String)
- Rename Source Object (page 256) (when the selected object is DN or Association and Enter String)
- Set Destination Attribute Value (page 263) (when the selected object is DN or Association and Enter Value Type is not structured)
- Set Destination Password (page 265)
- Set Local Variable (page 266)
- Set Operation Association (page 268)
- Set Operation Class Name (page 269)
- Set Operation Destination DN (page 270)
- Set Operation Property (page 271)
- Set Operation Source DN (page 272)
- Set Operation Template DN (page 273)
- Set Source Attribute Value (page 274)
- Set Source Password (page 276)
- Set XML Attribute (page 279)
- Status (page 280)
- Trace Message (page 285)

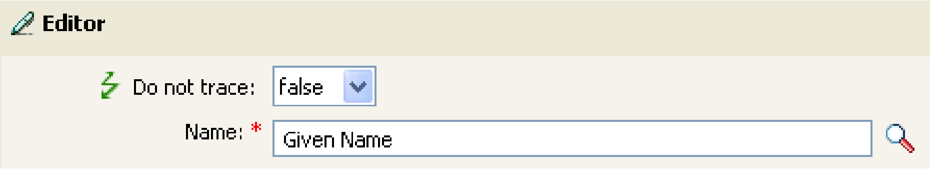
4.3.2 Argument Builder Example

The following example creates an argument for a user name 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.

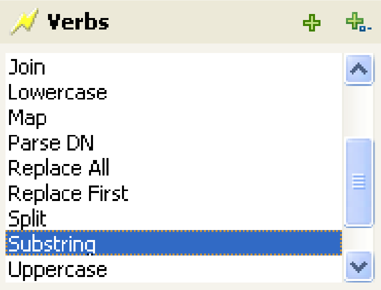


Editor

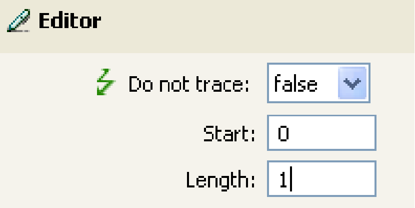
Do not trace: false

Name: * Given Name

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



4 Type 1 in the *Length* field.



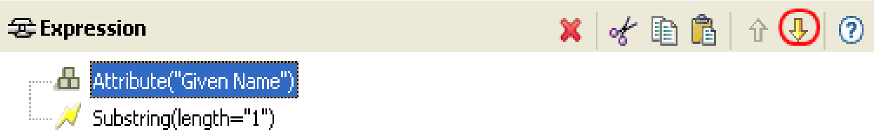
Editor

Do not trace: false

Start: 0

Length: 1

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



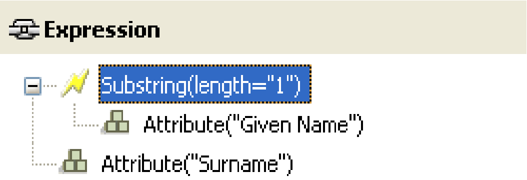
Expression

Attribute("Given Name")

Substring(length="1")

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

7 Specify or browse to the *Surname* attribute.



Expression

Substring(length="1")

Attribute("Given Name")

Attribute("Surname")

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 *OK* to save the argument.

4.4 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. Condition structures define the logic of condition groups. The two condition structures are:

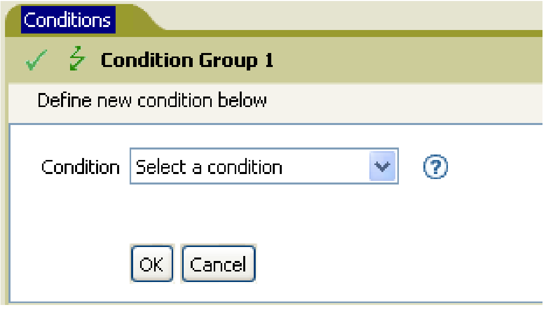
- *OR Conditions, AND Groups*
- *AND Conditions, OR Groups*

- Section 4.4.1, "Creating a Condition," on page 53
- Section 4.4.2, "Additional Options for the Condition Builder," on page 53

4.4.1 Creating a Condition

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

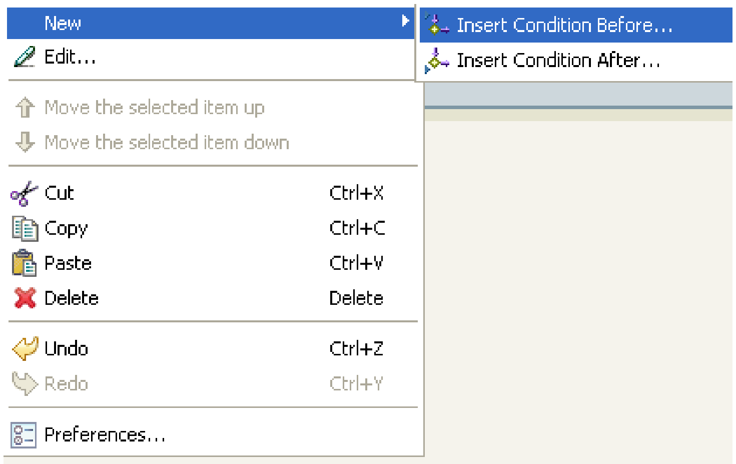
2 Double-click the Conditions tab to launch the Condition Builder.



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

4.4.2 Additional Options for the Condition Builder

1 Right-click the condition to see the additional options:


New
Edit...
Move the selected item up
Move the selected item down
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Undo Ctrl+Z
Redo Ctrl+Y
Preferences...
Insert Condition Before...
Insert Condition After...

- **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 the selected item up:** Moves the selected condition up in the order of execution.
- **Move the selected item down:** Moves the selected condition down in the order of execution.
- **Cut, Copy, Paste, or Delete:** Cuts, copies, pastes, or deletes the condition.
- **Undo or Redo:** Undoes or redoes the last action.
- **Preferences:** Allows you to set default functionary in the Policy builder.
- **Help:** Select a condition, then click the *Help* icon to see information specific to that condition.

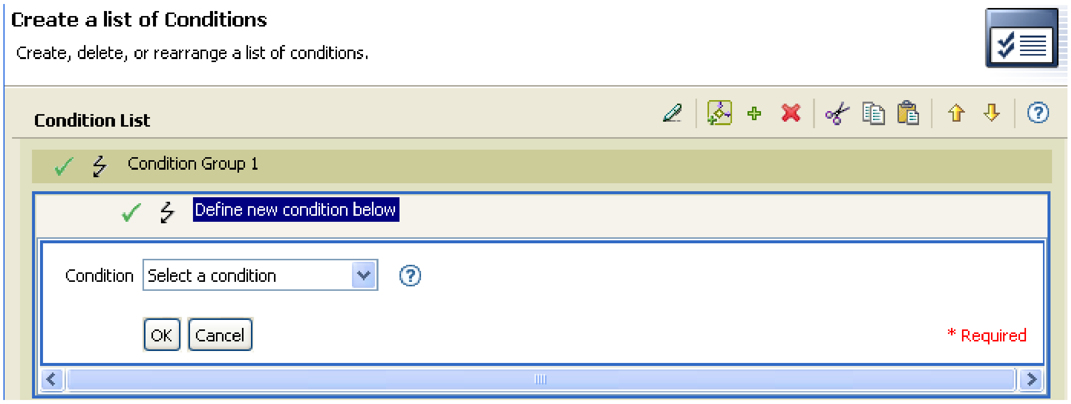
For additional information on the Condition Builder and the rules, see Section 3.4, "Creating a Rule," on page 32.

4.5 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 arguments* icon.

- If (page 243)
- While (page 289)

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


Create a list of Conditions

Create, delete, or rearrange a list of conditions.

2 Define the condition, then click *OK*.

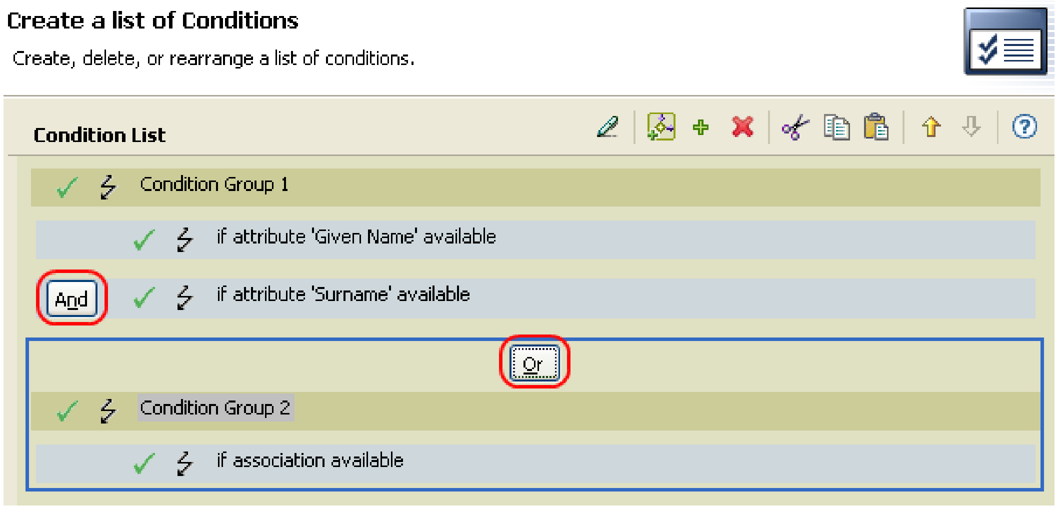
The Conditions builder allows for many different actions through the toolbar:

- The *Edit* icon opens the condition for edits.
- The *Append condition group* icon adds a new condition group. You can have multiple condition groups.
- The *Append new item* icon adds a new condition. You can have multiple conditions.
- The *Delete* icon deletes the selected condition or condition group.
- The *Cut*, *Copy*, and *Paste* icons allow you to use the clipboard to copy the conditions and condition groups.

- The *Move Up* and *Move Down* icons allow you to move the conditions and conditions groups.

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 4-4 *And/Or Icons in the Conditions Builder*



Create a list of Conditions

Create, delete, or rearrange a list of conditions.

Condition List

✓ Condition Group 1

✓ if attribute 'Given Name' available

And ✓ if attribute 'Surname' available

Or

✓ Condition Group 2

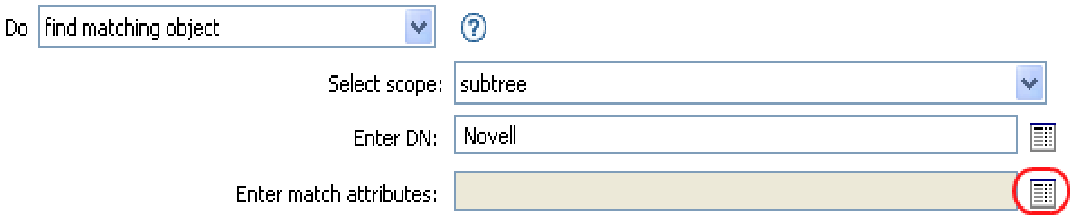
✓ if association available

4.6 Match Attribute Builder

The Match Attribute Builder enables you to select attributes and values used by the Find Matching Object (page 236) 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.



Do: find matching object

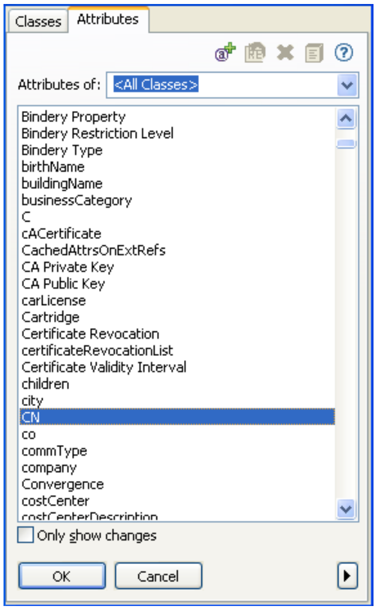
Select scope: subtree

Enter DN: Novell

Enter match attributes:

5. Click the *Browse attributes* icon to launch the Schema Browser.

6 Click the *Attributes* tab, then browse to and select the desired attribute.

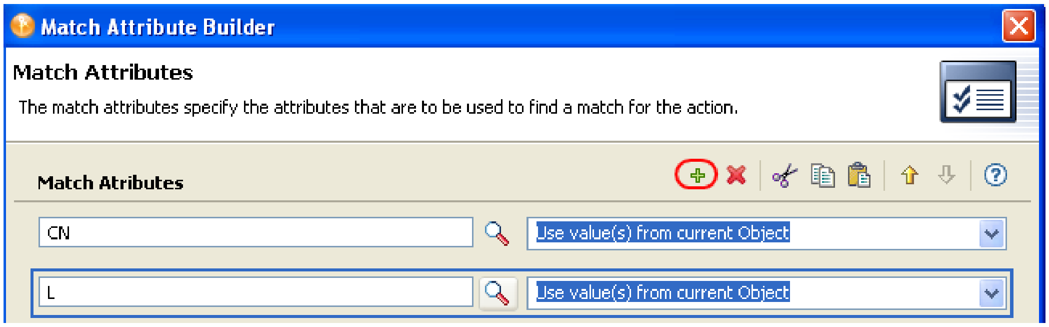


Attributes of: <All Classes>

☐ Only show changes

7 Click *OK*.

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



Match Attributes

The match attributes specify the attributes that are to be used to find a match for the action.

CN — Use value(s) from current Object

L — Use value(s) from current Object

8 Click *Finish*.

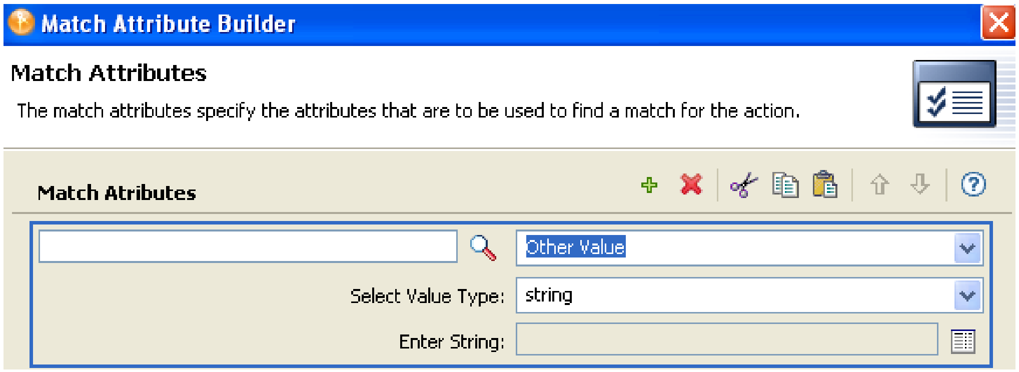
The Match Attribute Builder also allows you to specify another value, instead of using the value from the current object. Select *Other Value* instead of *Use values from current object*, to use a different value. There are multiple value types to specify:

- counter
- dn
- int
- interval
- octet
- state
- structured

- teleNumber
- time

To use the *Other Value*:

1 Launch the Match Attribute Builder, then select *Other Value*.



Match Attributes

The match attributes specify the attributes that are to be used to find a match for the action.

Match Atributes

Other Value

Select Value Type: string

Enter String:

2 Select the desired value type.

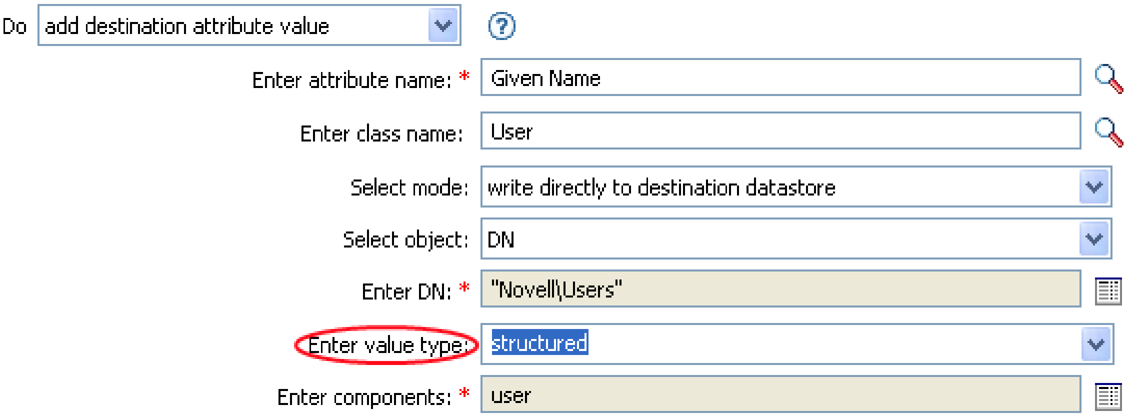
3 Specify the value, then click *OK*.

4.7 Action Argument Component Builder

To launch the Action Argument Component Builder, select one of the following actions when the *Enter value type* selection is *structured*, then click the *Edit components* icon ▤.

- Add Destination Attribute Value (page 216)
- Add Source Attribute Value (page 220)
- Reformat Operation Attribute Value (page 249)
- Remove Destination Attribute Value (page 252)
- Remove Source Attribute Value (page 253)
- Set Destination Attribute Value (page 263)
- Set Source Attribute Value (page 274)

***Figure 4-5** Add Destination Attribute Value Action*



Do add destination attribute value

Enter attribute name: * Given Name

Enter class name: User

Select mode: write directly to destination datastore

Select object: DN

Enter DN: * "Novell\Users"

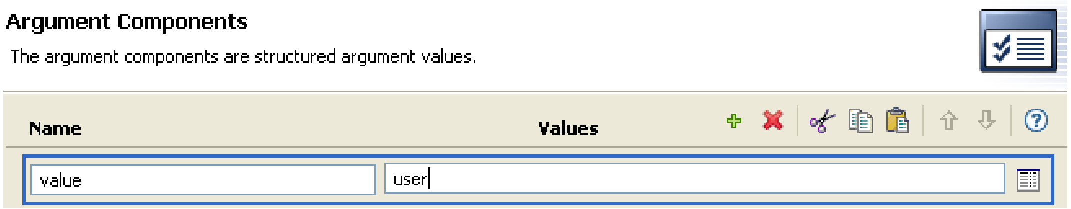
Enter value type: structured

Enter components: * user

1 Click the *Edit the components* icon ▤ when the value type is set to structured.

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.



Argument Components

The argument components are structured argument values.

Name	Values
value	user

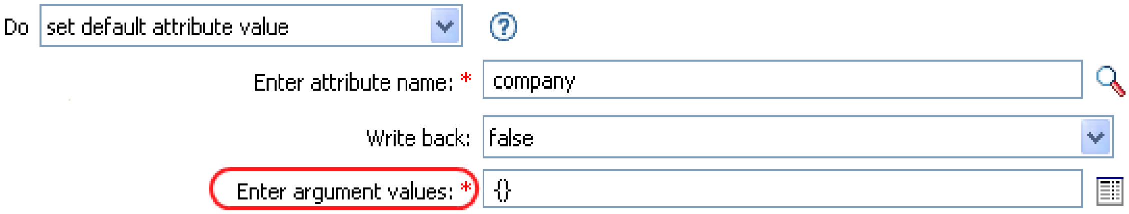
3 Click *Finish*.

4.8 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 261)

Figure 4-6 *Set Default Attribute Value*



Do: set default attribute value

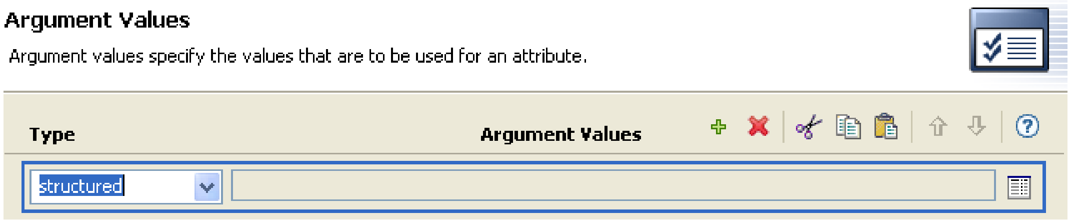
Enter attribute name: * company

Write back: false

Enter argument values: * {}

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

2 Click the *Edit the value lists* icon ▤.



Argument Values

Argument values specify the values that are to be used for an attribute.

Type	Argument Values
structured	

3 Click the *Edit the arguments* icon ▤.

4 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.

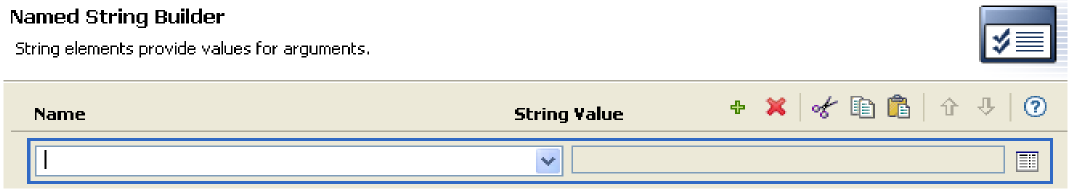
5 Click *Finish*.

4.9 Named String Builder

To launch the Named String Builder, select one of the following actions, then click the *Edit the strings* icon ▤.

- Generate Event (page 240)
- Send Email (page 257)
- Send Email from Template (page 259)

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.

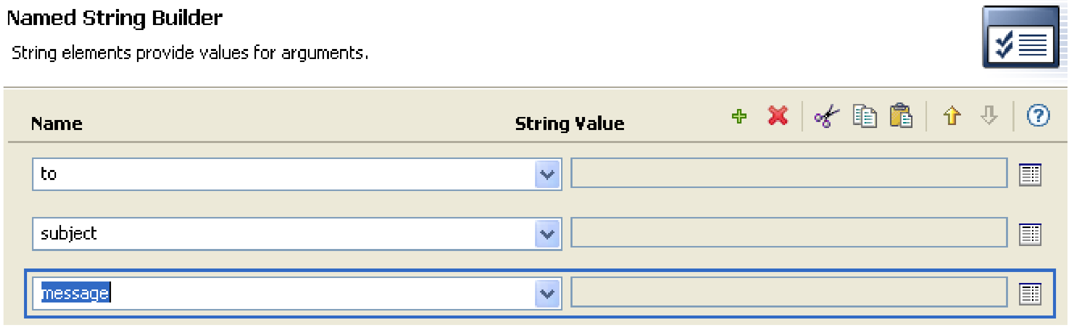


Named String Builder

String elements provide values for arguments.

3. Click *Finish*.

   For a Send Email action, the named strings correspond to the elements of the e-mail:



Named String Builder

String elements provide values for arguments.

to

subject

message

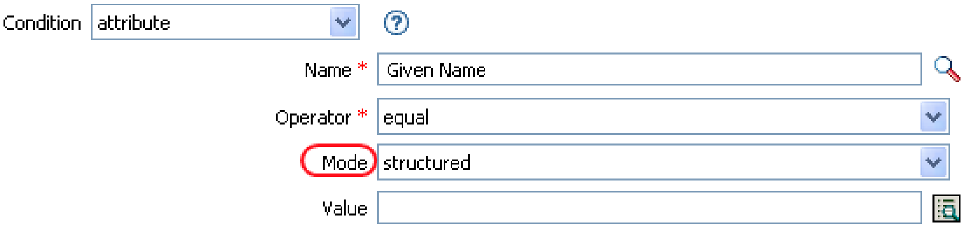
A complete list of possible values is contained in the help file corresponding to the action that launches the Named String Builder.

4.10 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 172)
- If Destination Attribute (page 178)
- If Association (page 170)

Figure 4-7 If Attribute mode



Condition: attribute

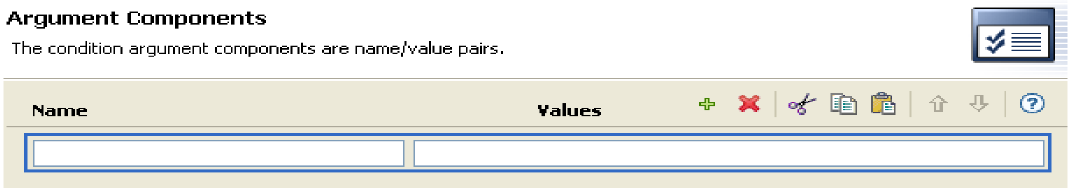
Name *: Given Name

Operator *: equal

Mode: structured

Value:

1 Specify the name and value of the condition component.



Argument Components

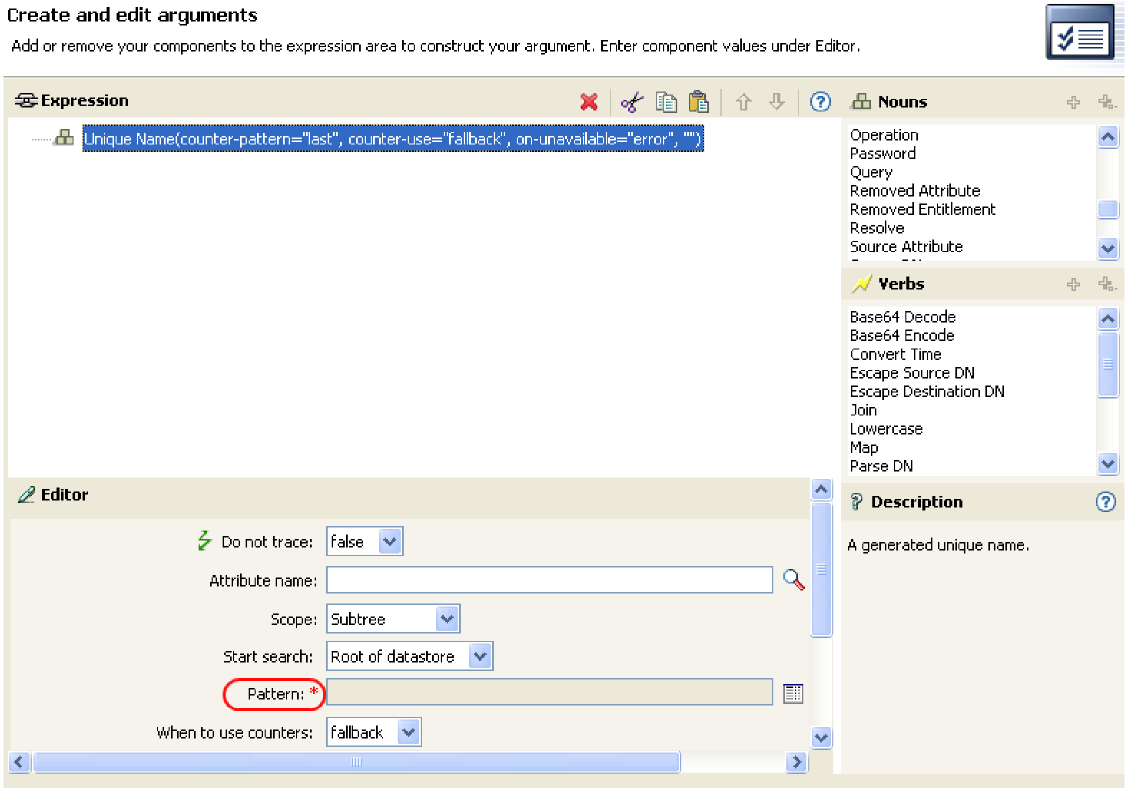
The condition argument components are name/value pairs.

2 Click *Finish*.

4.11 Pattern String Builder

You can launch the Pattern String Builder from the Argument Builder editor when the Unique Name (page 325) token is selected. The Argument Builder editor pane shows a Pattern field where you can click to launch the Pattern String Builder.

Figure 4-8 Unique Name Token in the Argument Builder



Create and edit arguments

Add or remove your components to the expression area to construct your argument. Enter component values under Editor.

Expression

Unique Name(counter-pattern="last", counter-use="fallback", on-unavailable="error", "")

Nouns

Operation
Password
Query
Removed Attribute
Removed Entitlement
Resolve
Source Attribute

Verbs

Base64 Decode
Base64 Encode
Convert Time
Escape Source DN
Escape Destination DN
Join
Lowercase
Map
Parse DN

Editor

Do not trace: false

Attribute name:

Scope: Subtree

Start search: Root of datastore

Pattern: *

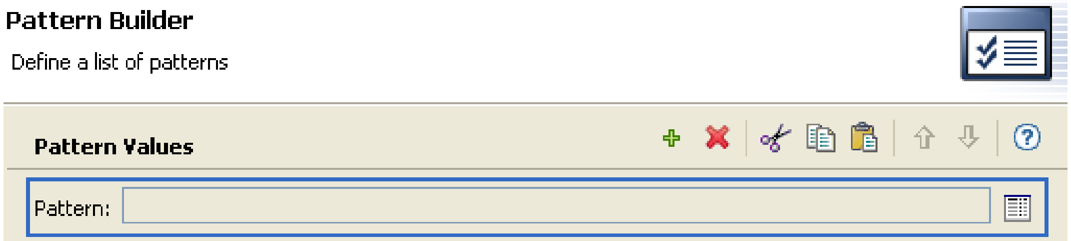
When to use counters: fallback

Description

A generated unique name.

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.



Pattern Builder

Define a list of patterns

Pattern Values

Pattern:

3 Click *Finish*.

4.12 String Builder

The String Builder enables you to construct name/value pairs for use in certain actions, such as Start Workflow.

1 Specify the name of the string in the *Name* field.

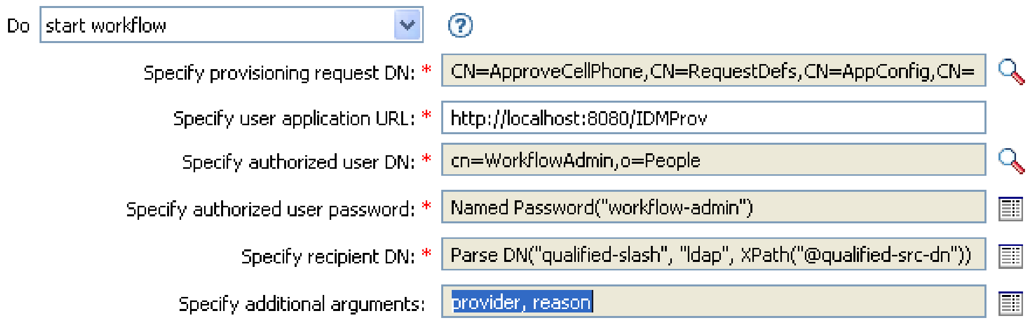
2 Click the Edit the arguments icon to construct the value of the string.

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

The Start Workflow action contains an *Additional Arguments* field that allows you to create the necessary values for a specific workflow. These fields are unique to each workflow. The string builder allows you to create strings and set values for these strings.

The following example starts a workflow process each there is an add operation. The workflow is a request for a cell phone. The strings in the Additional Arguments field are the cell phone provider and the reason the cell is requested.

Figure 4-9 *Start Workflow Example*



Do: start workflow

Specify provisioning request DN: * CN=ApproveCellPhone,CN=RequestDefs,CN=AppConfig,CN=

Specify user application URL: * http://localhost:8080/IDMProv

Specify authorized user DN: * cn=WorkflowAdmin,o=People

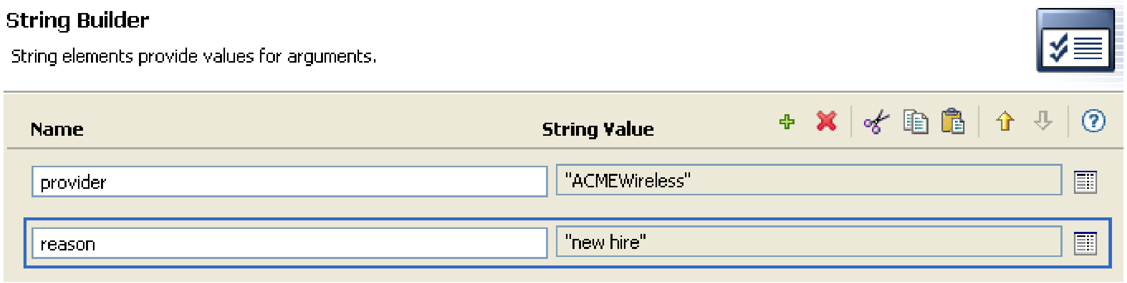
Specify authorized user password: * Named Password("workflow-admin")

Specify recipient DN: * Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))

Specify additional arguments: provider, reason

When you click the icon to the right of the *Additional Arguments* field, the String Builder is launched. In this example, the value for the provider is set to ACMEWireless, and the value for the reason is set to new hire. To view this example in XML, see start_workflow.xml (../samples/start_workflow.xml).

Figure 4-10 *String Builder Example*



String Builder

String elements provide values for arguments.

Name	String Value
provider	"ACMEWireless"
reason	"new hire"

4.13 XPath Builder

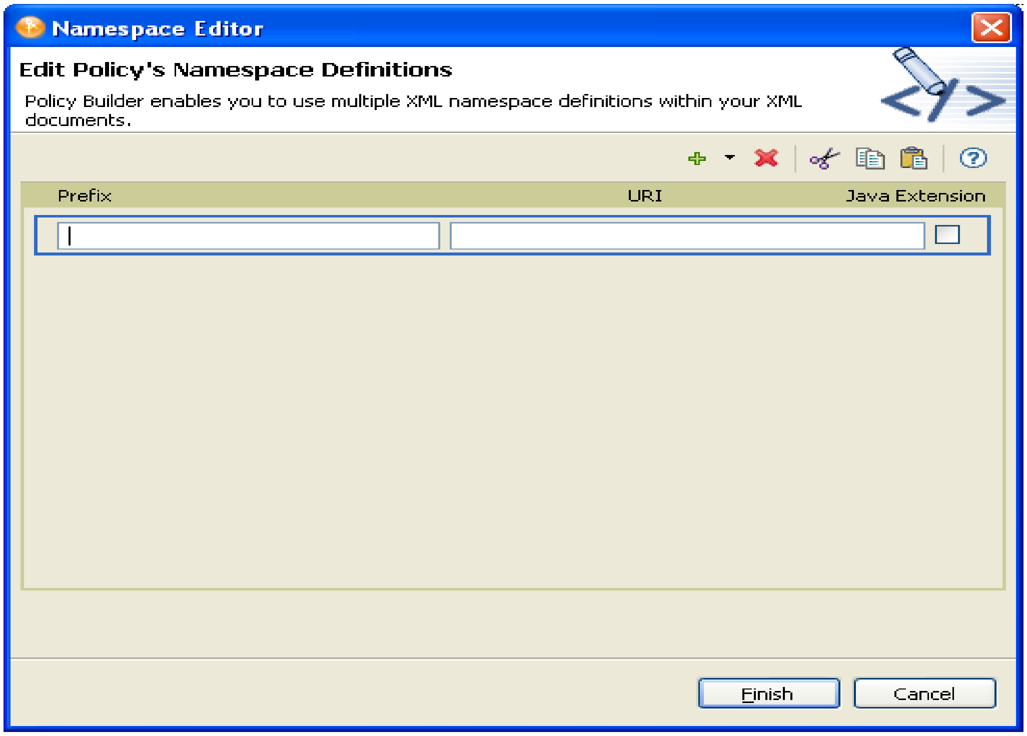
The XPath Builder is a powerful tool that allows you to build and test an XPath expression against any XML document. See "Using the XPath Builder" on page 65 for more information.

4.14 Namespace Editor

The Policy Builder enables you to use multiple XML namespaces within your XML documents. To define a namespace, specify the namespace prefix in the *Name* field, and the URI in the *URI* field. Leave the *Java Extension* check box deselected.

You can also access Java* classes through XPath 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.

Figure 4-11 Namespace Editor



Namespace Editor
Edit Policy's Namespace Definitions
Policy Builder enables you to use multiple XML namespace definitions within your XML documents.
Prefix
URI
Java Extension
Finish
Cancel

4.14.1 Accessing Java Classes Using Namespaces

Novell provides several Identity Manager Java classes that can be called using XPath expressions from the Policy Builder. The following links open Javadoc references for these Java classes:

- com.novell.nds.dirxml.driver.XdsQueryProcessor (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/XdsQueryProcessor.html)
- com.novell.nds.dirxml.driver.XdsCommandProcessor (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/XdsCommandProcessor.html)
- com.novell.nds.dirxml.driver.DNConverter (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/DNConverter.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.

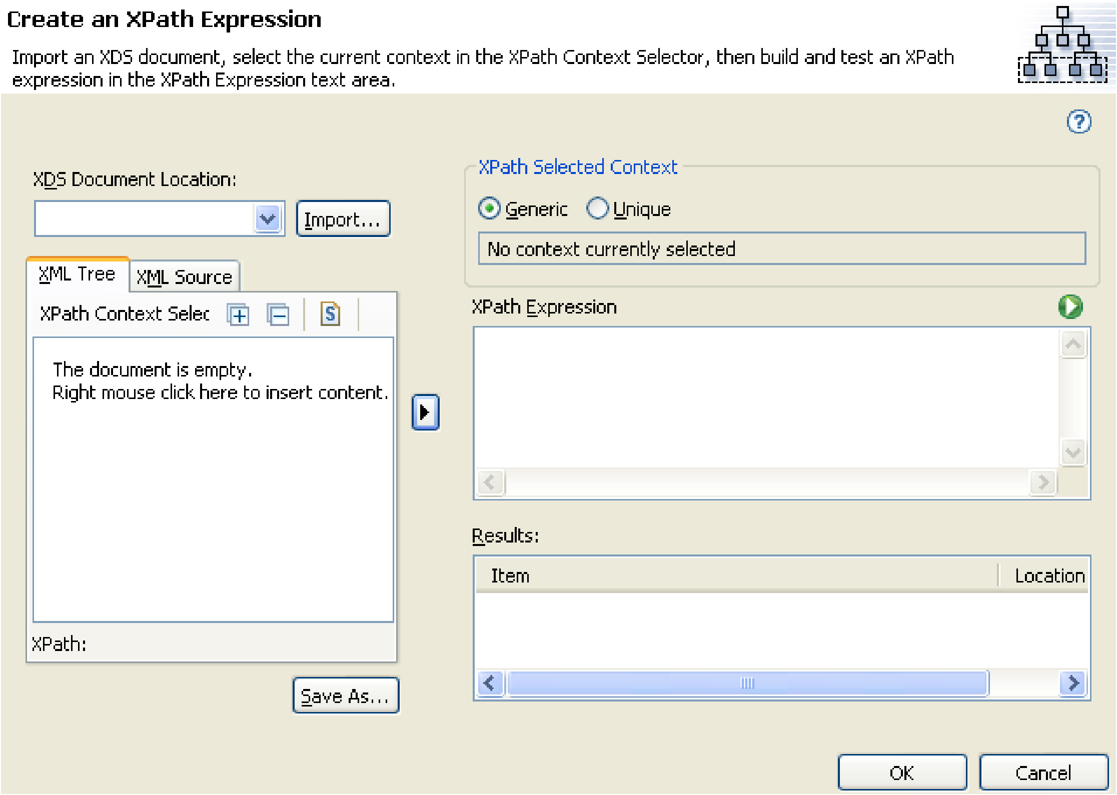
For additional information on using XPath and the Novell Java classes listed above, consult the DirXML Driver Developer Kit (http://developer.novell.com/documentation/dirxml/dirxmlbk/ref/dirxmlfaq.html).

Using the XPath Builder

5

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 *Understanding Policies for Identity Manager 3.5.1*.

Figure 5-1 *XPath Builder*



Create an XPath Expression

Import an XDS document, select the current context in the XPath Context Selector, then build and test an XPath expression in the XPath Expression text area.

XDS Document Location:

XPath Selected Context

● Generic ○ Unique

No context currently selected

The document is empty.
Right mouse click here to insert content.

XPath:

XPath Expression

Results:

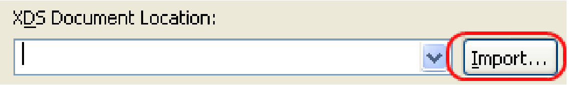
Item	Location

To use the XPath Builder:

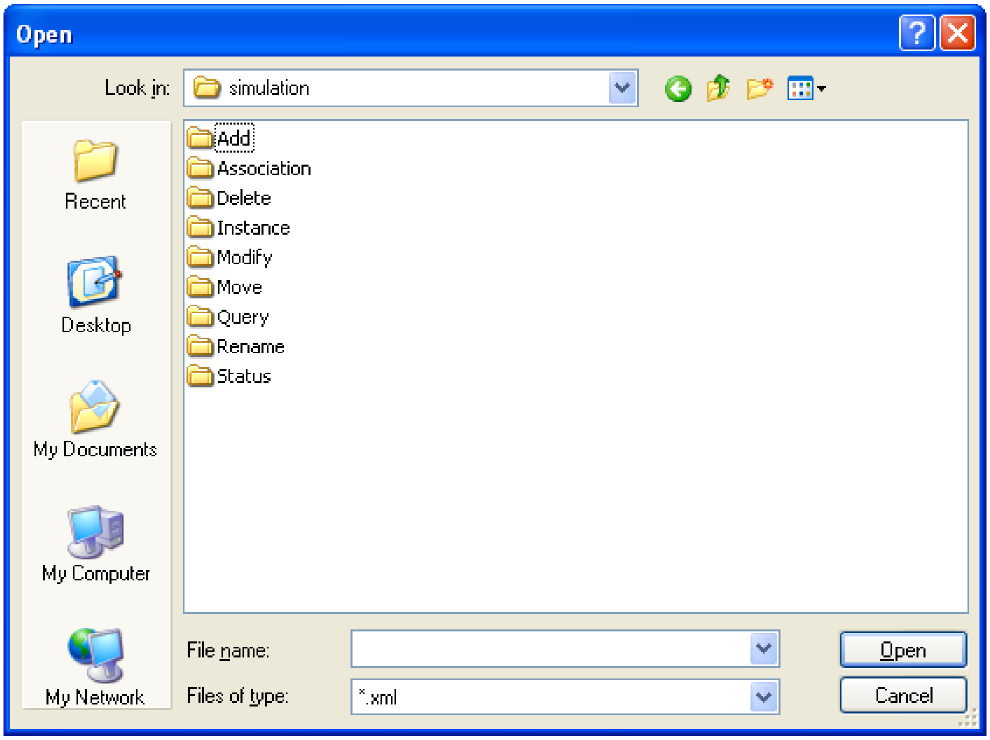
1 In the Policy Builder, select any of the following conditions or action, then click the *Launch XPath Builder* icon .

- If XML Attribute (page 207)
- If XPath Expression (page 209)
- Append XML Element (page 222)
- Append XML Text (page 224)
- Clone By XPath Expression (page 231)
- Set XML Attribute (page 279)
- Strip XPath (page 284)

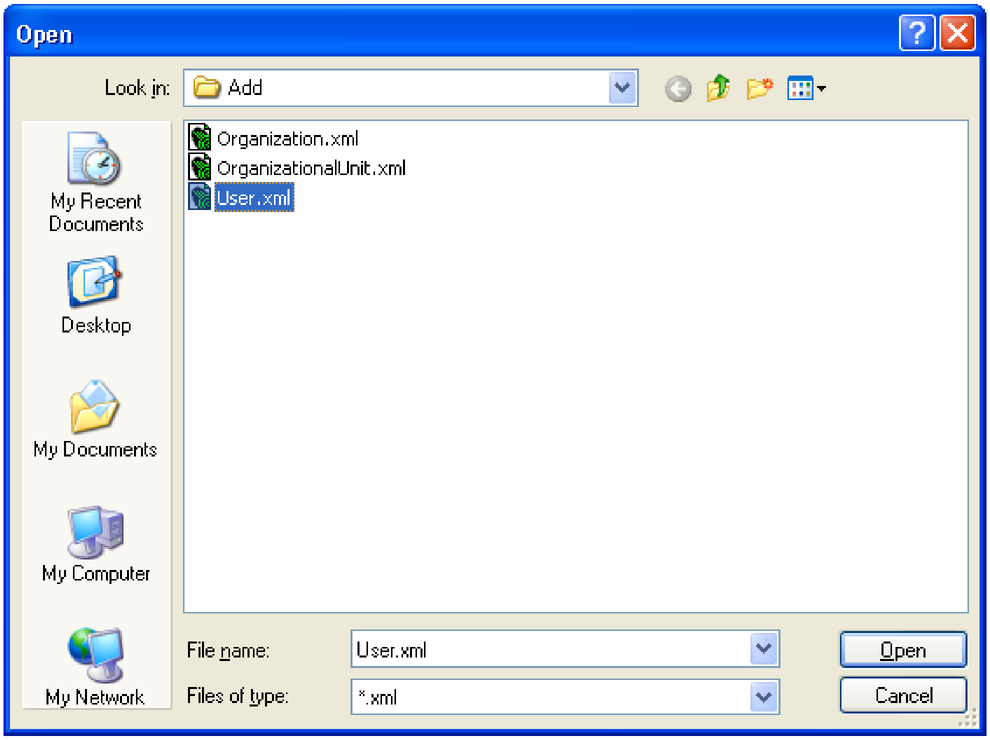
2 Select *Import* to browse to and select the XDS document to test.



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.

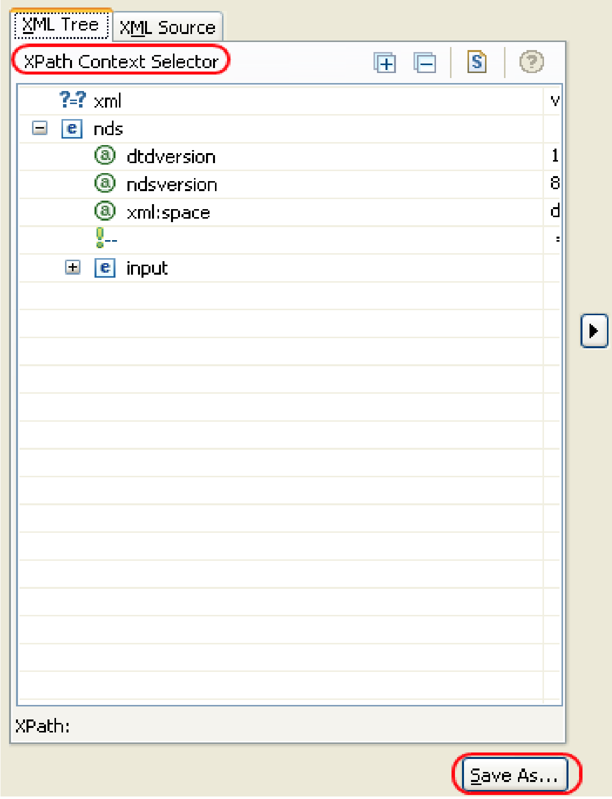


File name: User.xml

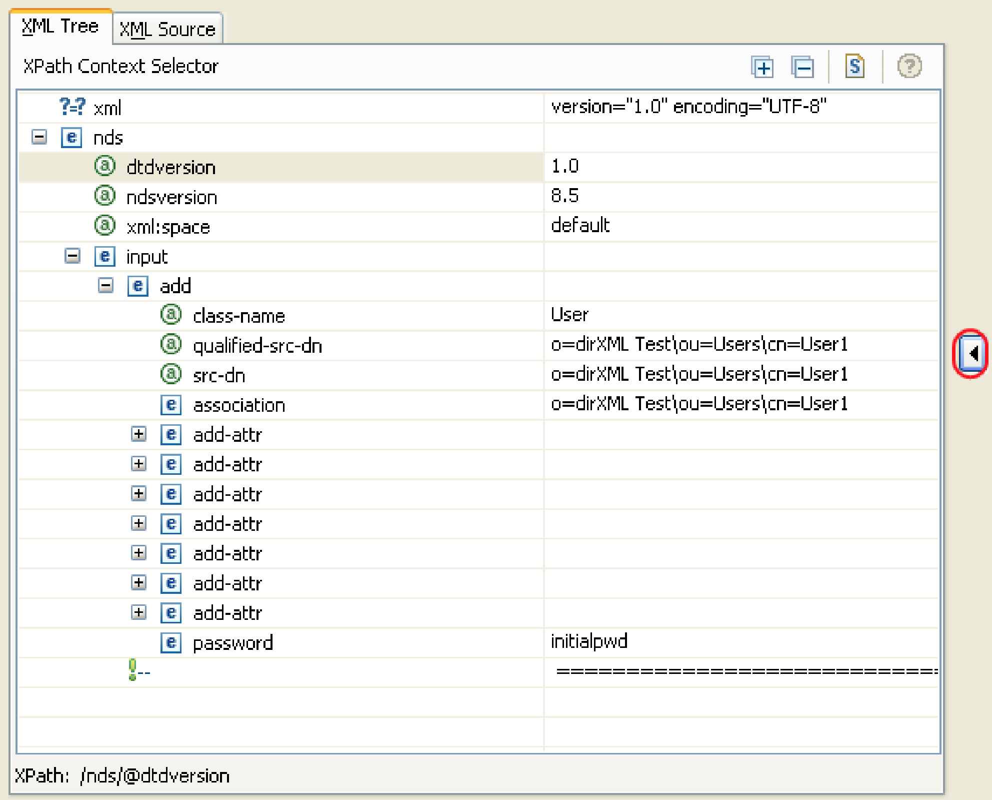
Files of type: *.xml

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.



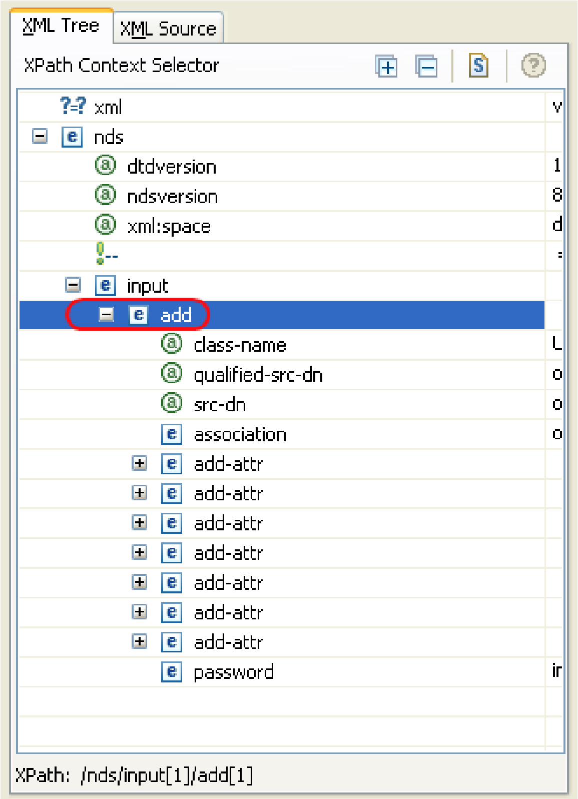
If you want to see the XDS document without scrolling, click the *Hide XPath Details* icon ■. To see the *XPath Expression* and *Results* windows, click *Show XPath Details* icon ■.



?=? xml	version="1.0" encoding="UTF-8"
⊟ e nds	
ⓐ dtdversion	1.0
ⓐ ndsversion	8.5
ⓐ xml:space	default
⊟ e input	
⊟ e add	
ⓐ class-name	User
ⓐ qualified-src-dn	o=dirXML Test\ou=Users\cn=User1
ⓐ src-dn	o=dirXML Test\ou=Users\cn=User1
e association	o=dirXML Test\ou=Users\cn=User1
⊞ e add-attr	
⊞ e add-attr	
⊞ e add-attr	
⊞ e add-attr	
⊞ e add-attr	
⊞ e add-attr	
⊞ e add-attr	
e password	initialpwd
!--	============================

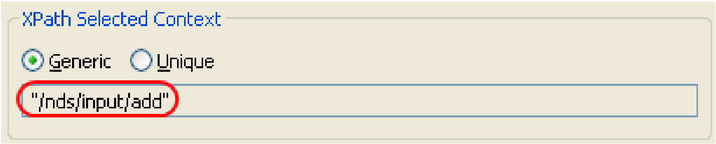
XPath: /nds/@dtdversion

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



XPath: /nds/input[1]/add[1]

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



● Generic ○ Unique

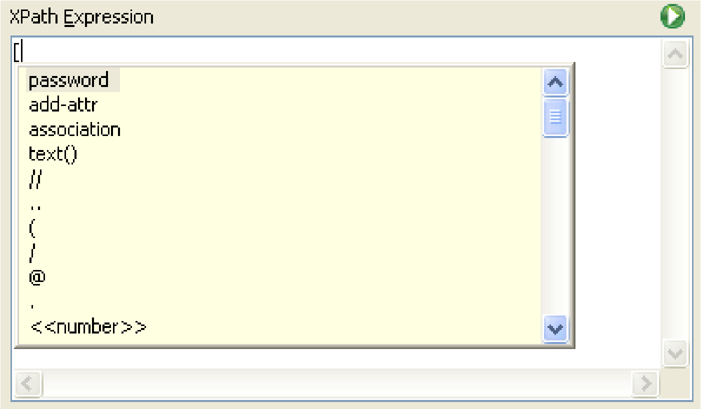
"/nds/input/add"

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.

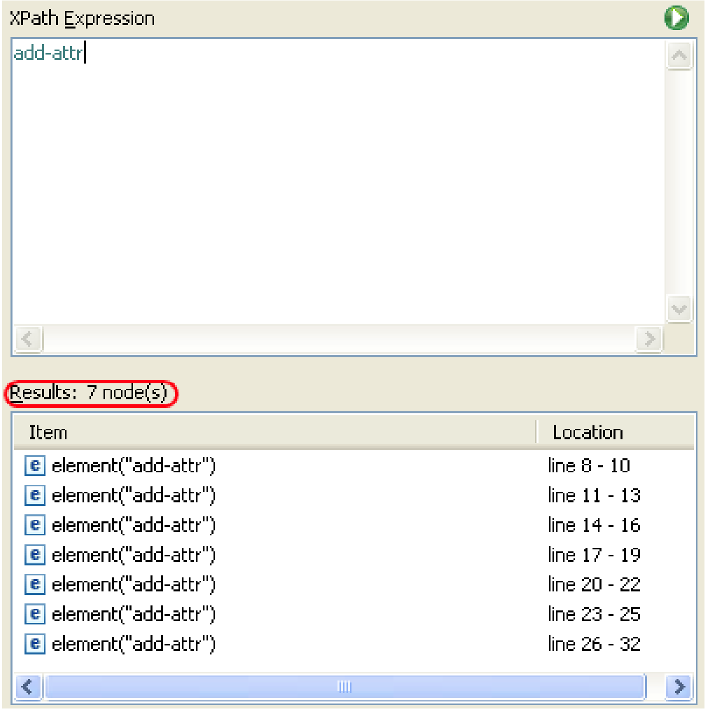
Unique searches the XML document until it finds a match and stops. The unique XPath expression is "/nds/input[1]/add[1]". It searches for the first instance of add and stops. You can specify which instance you want to use by selecting the next instance of the XPath element in the *XML Context Selector*.

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.



```
add-attr
```

Results: 7 node(s)

Item	Location
element("add-attr")	line 8 - 10
element("add-attr")	line 11 - 13
element("add-attr")	line 14 - 16
element("add-attr")	line 17 - 19
element("add-attr")	line 20 - 22
element("add-attr")	line 23 - 25
element("add-attr")	line 26 - 32

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 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.

Defining Schema Mapping Policies

6

Schema Mapping policies map class names and attribute names between the Identity Vault namespace and the application namespace. The same schema mapping policy is applied in both directions. All documents that are passed in either direction on either channel between the Metadirectory engine and the application shim are passed through the Schema Mapping policy.

There is one Schema Mapping policy per driver.

- Section 6.1, “Accessing the Schema Map Editor,” on page 73
- Section 6.2, “Editing a Schema Mapping Policy,” on page 77
- Section 6.3, “Testing Schema Mapping Policies,” on page 79
- Section 6.4, “Accessing the Schema Mapping Policy in XML,” on page 82
- Section 6.5, “Additional Schema Map Policy Options,” on page 82

6.1 Accessing the Schema Map Editor

The Schema Map editor allows you to edit the Schema Mapping policies. There are three different ways to access the Schema Map editor in Designer: through the Outline view, through the Policy Flow view, or through the Policy Set view.

- Section 6.1.1, “Outline View,” on page 73
- Section 6.1.2, “Policy Flow View,” on page 74
- Section 6.1.3, “Policy Set View,” on page 75
- Section 6.1.4, “Keyboard Support,” on page 76

6.1.1 Outline View

1 In an open project, click the *Outline* tab.

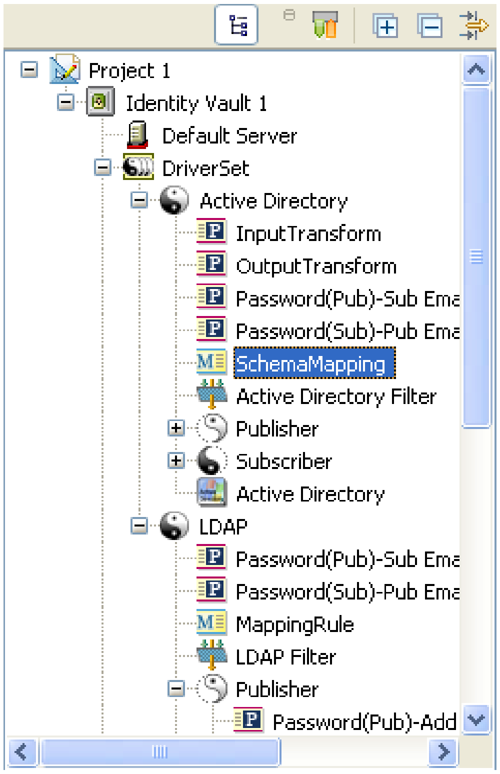
2 Click the *Show Model Outline* icon.

3 Select the driver you want to manage the schema mapping policy on, and click the plus sign to the right.

4 Double-click the Schema Map icon to launch the Schema Map editor.

or

Right-click and select *Edit*.

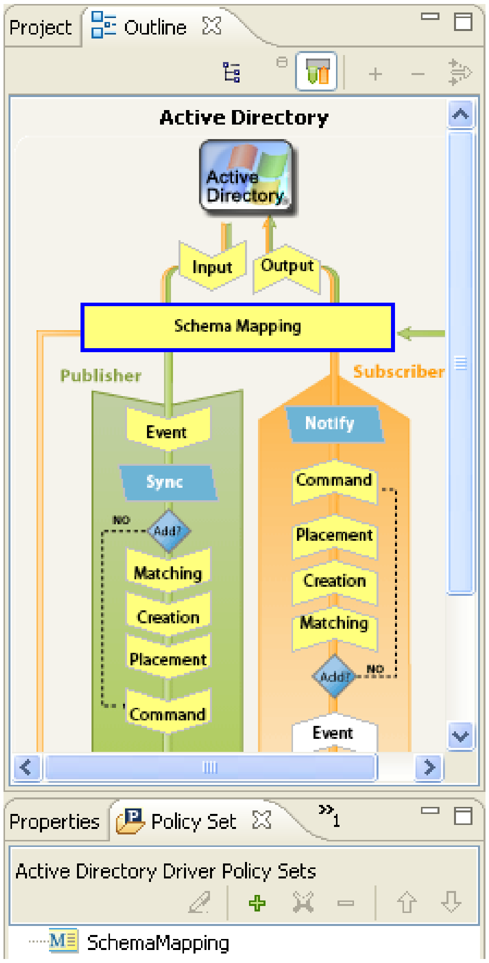


6.1.2 Policy Flow View

1. In an open project, click the *Outline* tab.
2. Click the *Show Policy Flow* icon.
3. Double-click the Schema Mapping policy to launch the Schema Map editor.

   or

Right-click and select *Edit Policy* to launch the Schema Map editor.



Active Directory Driver Policy Sets

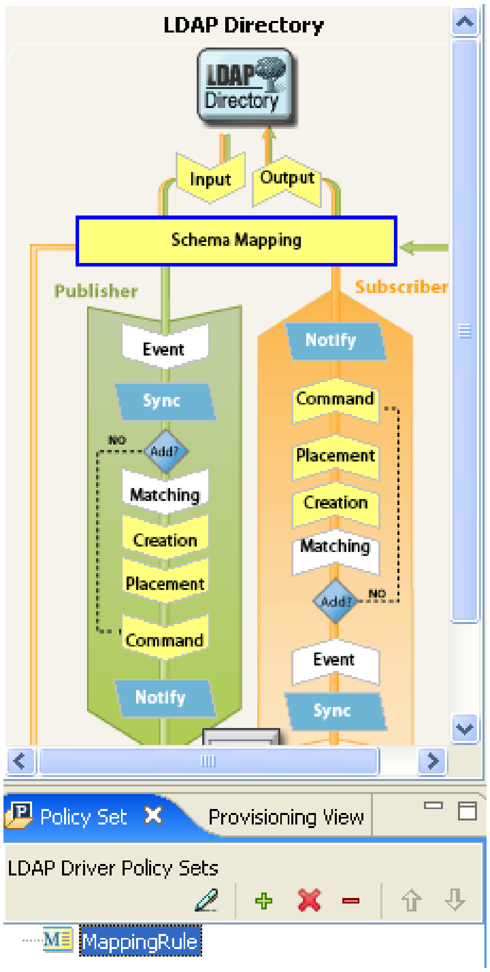
SchemaMapping

6.1.3 Policy Set View

1 Double-click the Schema Map policy in the Policy Set view.

or

Right-click the Schema Map policy and select *Edit*.



6.1.4 Keyboard Support

Table 6-1 *Schema Map Editor Keyboard Support*

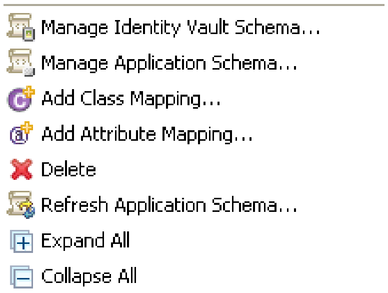
Action	Description
Up-arrow	Moves the cursor up in the Schema Map editor.
Down-arrow	Moves the cursor down in the Schema Map editor.
Left-arrow	Collapses the information displayed
Right-arrow	Expands the information displayed.
Insert	Adds a class.
Ctrl+Insert	Adds an attribute.
Delete	Deletes the selected items.
Enter	Accesses the edit mode. Press Enter a second time to save the changes.

Action	Description
Esc	Exits the edit mode.

6.2 Editing a Schema Mapping Policy

The Schema Map editor allows you to create and edit schema mapping policies. To display a context menu, right-click an item.

Figure 6-1 *Context Menu of the Schema Map Editor*


Manage Identity Vault Schema...
Manage Application Schema...
Add Class Mapping...
Add Attribute Mapping...
Delete
Refresh Application Schema...
Expand All
Collapse All

- Section 6.2.1, "Removing or Adding Classes and Attributes," on page 77
- Section 6.2.2, "Refreshing the Application Schema," on page 78
- Section 6.2.3, "Editing Items," on page 79
- Section 6.2.4, "Sorting Items," on page 79
- Section 6.2.5, "Managing the Schema," on page 79

6.2.1 Removing or Adding Classes and Attributes

- "Removing a Class or Attribute" on page 77
- "Adding a Class" on page 78
- "Adding an Attribute" on page 78

Removing a Class or Attribute

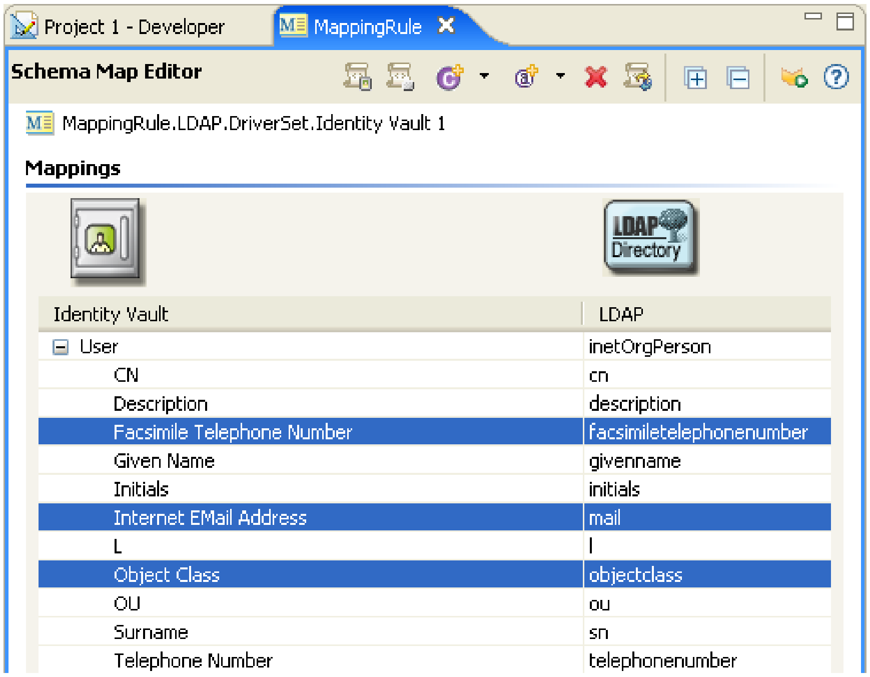
If you do not want a class or an attribute to be mapped to a class or attribute in the connected system, the best practice is to completely remove the class or the attribute from the Schema Mapping policy. There are three different ways to add or remove attributes and classes from the Schema Mapping policy:

- Select the class or attribute you want to remove, then right-click and click *Delete*.
- Select the class or attribute you want to remove, then click the *Delete* icon ✖ in the upper right corner.
- Select the class or attribute you want to remove, then press the Delete key.

You can select multiple classes or attributes to delete at the same time.

1 Press Ctrl and select each item with the mouse.

2 Press the Delete key to delete the items.



Identity Vault	LDAP
⊟ User	inetOrgPerson
CN	cn
Description	description
Facsimile Telephone Number	facsimiletelephonenumber
Given Name	givenname
Initials	initials
Internet EMail Address	mail
L	l
Object Class	objectclass
OU	ou
Surname	sn
Telephone Number	telephonenumber

Adding a Class

1 Right-click in the Schema Map editor, then click *Add Class Mapping*.

or

Select the *Add Class Mapping* icon in the upper right corner.

2 From the drop-down list for the Identity Vault, select the class you want to add.

3 From the drop-down list for the connected system, select the class you want to map to.

4 To save the changes, click *File* > *Save*.

Adding an Attribute

1 Right-click in the Schema Map editor, then click *Add Attribute Mapping*.

or

Select the *Add Attribute Mapping* icon in the upper right corner.

2 From the drop-down list for the Identity Vault, select the attribute you want to add.

3 From the drop-down list for the connected system, select the attribute you want to map to.

4 To save the changes, click *File* > *Save*.

6.2.2 Refreshing the Application Schema

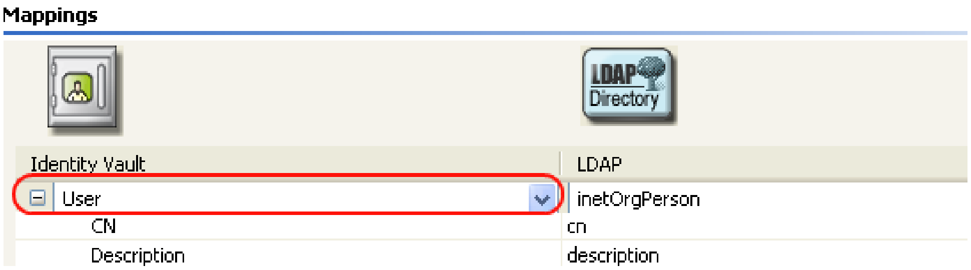
If you have modified the schema in the application, these changes need to be reflected in the Schema Mapping policy. To make the new schema available, click the *Refresh application schema* icon in the toolbar.

When you create a new class or attribute mapping, you can see the new schema in the drop-down list for the connected application.

6.2.3 Editing Items

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

Figure 6-2 *Schema Map Editor*



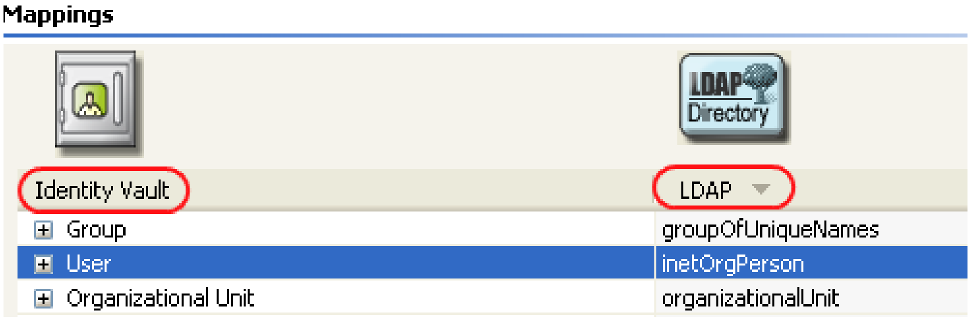
Mappings

Identity Vault	LDAP
⊟ User	inetOrgPerson
CN	cn
Description	description

6.2.4 Sorting Items

The Schema editor allows you to sort the items in ascending order based on either Identity Manager or the connected system. To sort, click the header of either column.

Figure 6-3 *Schema Map Editor Sorting Items*



Mappings

Identity Vault	LDAP ▼
⊞ Group	groupOfUniqueNames
⊞ User	inetOrgPerson
⊞ Organizational Unit	organizationalUnit

6.2.5 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 connected systems. To manage the Identity Vault schema, right-click in the Schema Map editor and click *Manage Identity Vault Schema*. To manage the connected systems schema, right-click in the Schema Map editor and click *Manage Application Schema*. For more information about how to manage the schema, see "Managing the Schema" in the *Designer 2.1 for Identity Manager 3.5.1*.

6.3 Testing Schema Mapping Policies

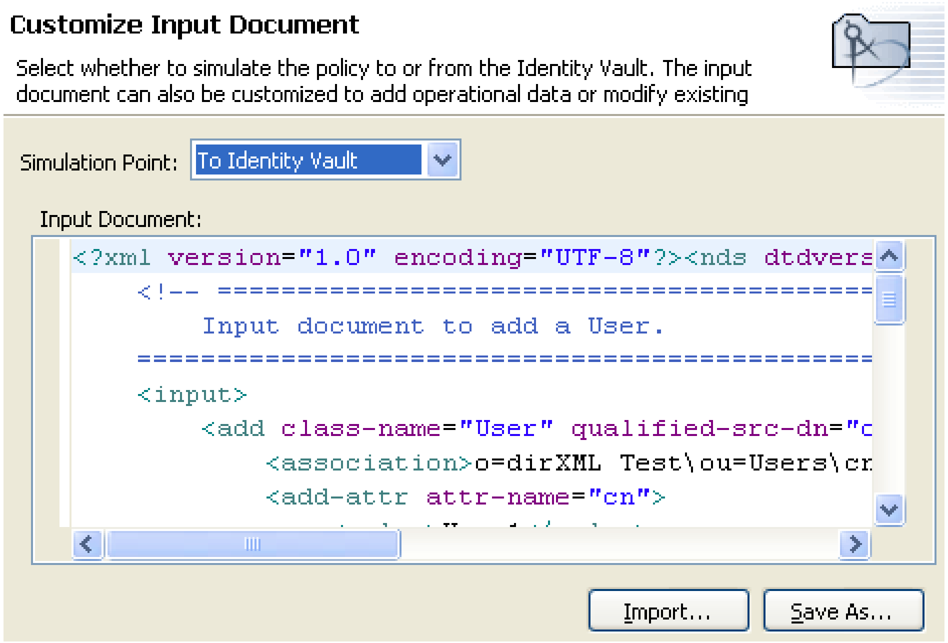
Designer comes with a tool called the Policy Simulator. It allows you to test your policies without implementing them in a production environment. You can launch the Policy Simulator through the Schema Mapping editor to test your policy after you have modified it.

To access the Policy Simulator and test the Schema Mapping 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.
3. Select *Import* to browse to a file that simulates an event.
4. Select the file, then click *Open*.

   This example uses the `com.novell.designer.idm.policy\simulation\add\user.xml` file, which simulates an Add event for a user object.

   The Policy Simulator displays the input document of the user Add event.
5. Click *Next* to begin the simulation.



Customize Input Document

Select whether to simulate the policy to or from the Identity Vault. The input document can also be customized to add operational data or modify existing

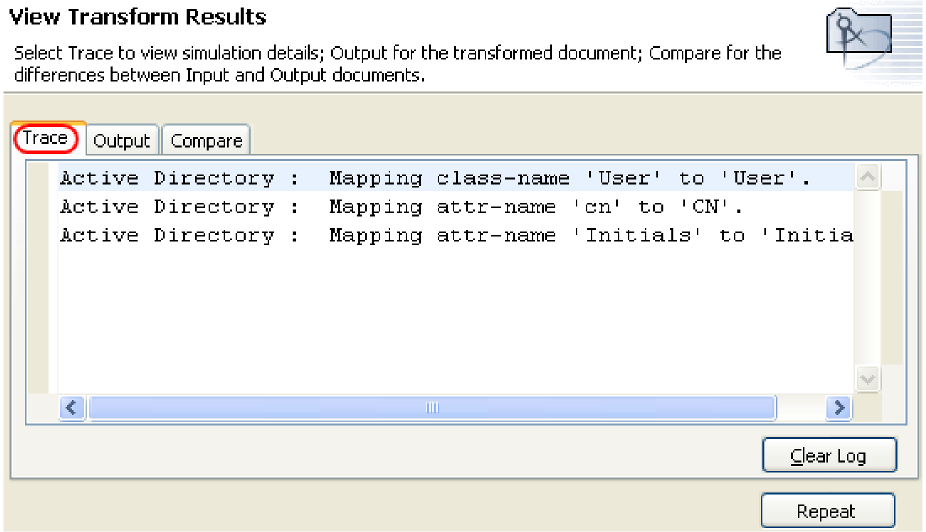
Simulation Point: To Identity Vault

Input Document:

```
<?xml version="1.0" encoding="UTF-8"?><nds dtdvers
    <!-- ==========================================
        Input document to add a User.
    ============================================
    <input>
        <add class-name="User" qualified-src-dn="c
            <association>o=dirXML Test\ou=Users\cn
            <add-attr attr-name="cn">
```

The Policy Simulator displays the log of the Add event, the output document, and a comparison of the input document to the output document that was generated.

6 Select the *Trace* tab to see the results of the Add event as you would through DSTRACE.



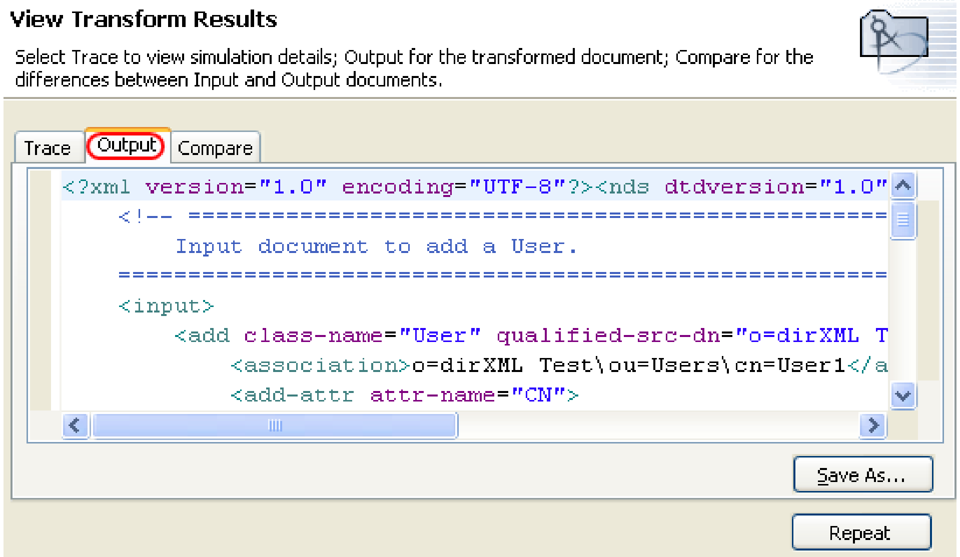
View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output documents.

```
Active Directory :  Mapping class-name 'User' to 'User'.
Active Directory :  Mapping attr-name 'cn' to 'CN'.
Active Directory :  Mapping attr-name 'Initials' to 'Initia
```

Click *Clear Log*, then click *Repeat* to run the simulation again with new trace log.

7 Select the *Output* tab to view the output document that is generated from the Schema Map policy executed against the input document. In this example, it is the user Add event.



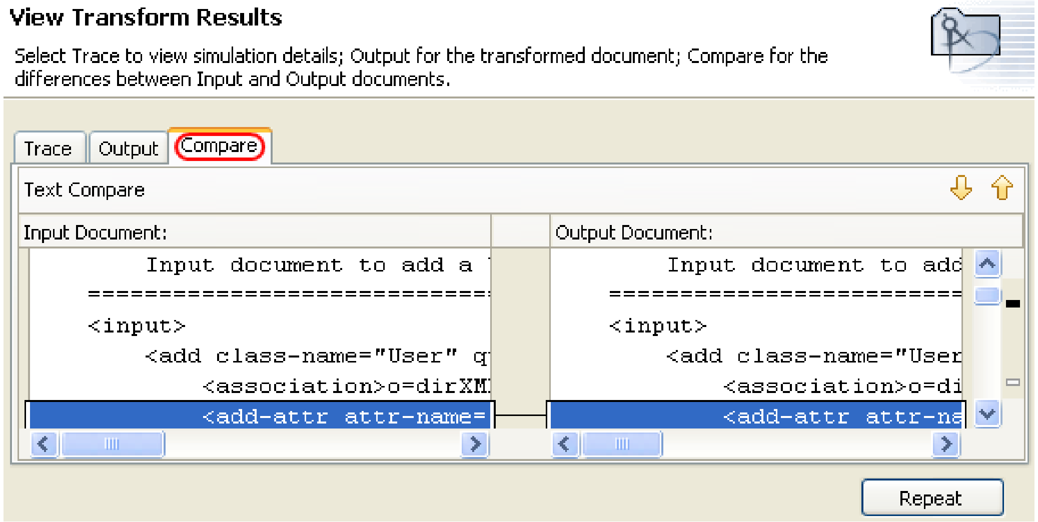
View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output documents.

```
<?xml version="1.0" encoding="UTF-8"?><nds dtdversion="1.0"
    <!-- ===================================================
        Input document to add a User.
    ========================================================
    <input>
        <add class-name="User" qualified-src-dn="o=dirXML T
            <association>o=dirXML Test\ou=Users\cn=User1</a
            <add-attr attr-name="CN">
```

You can edit the input and output document, then click *Save As* to save the output to an XML file.

8 Select the *Compare* tab to compare the text of the input document to the document that is generated, which is the output document.



View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output documents.

```
        Input document to add a
    ============================
    <input>
        <add class-name="User" q
            <association>o=dirXM
            <add-attr attr-name=
```

```
        Input document to add
    ========================
    <input>
        <add class-name="User
            <association>o=di
            <add-attr attr-na
```

9 Click *Repeat* to select a different input document and see the results of that event.

10 When you have finished testing the Schema Mapping Policy, click *Finish* to close the Policy Simulator.

6.4 Accessing the Schema Mapping 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 Novell XML Editor” in the *Designer 2.1 for Identity Manager 3.5.1*.

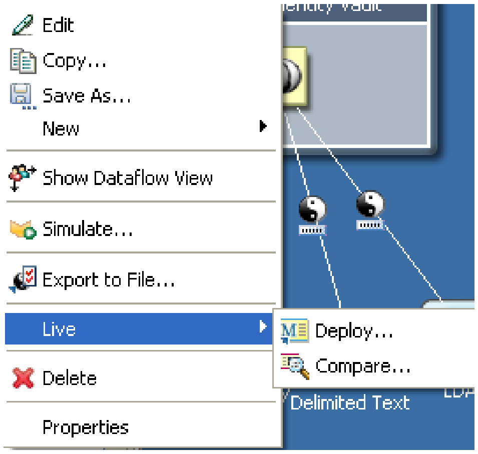
6.5 Additional Schema Map Policy Options

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.

- Section 6.5.1, “Outline View Additional Options,” on page 83
- Section 6.5.2, “Policy Flow View Additional Options,” on page 84
- Section 6.5.3, “Policy Set View Additional Options,” on page 85

6.5.1 Outline View Additional Options

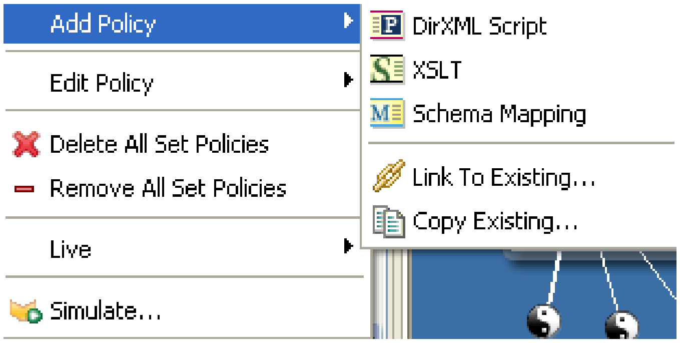
1 Right-click the Schema Map policy in the Outline view.



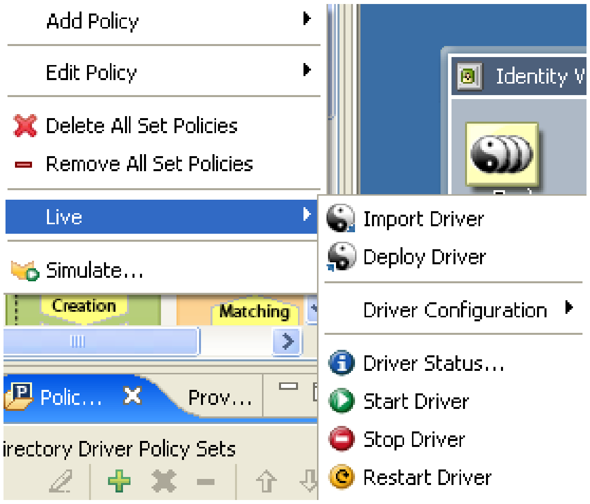
- **Edit:** Launches the Schema Map editor. For more information, see Chapter 6, “Defining Schema Mapping Policies,” on page 73.
- **Copy:** Creates a copy of the Schema Map policy.
- **Save As:** Saves the Schema Map policy as a `.xml` file.
- **New:** Creates a Domain Group or an Identity Vault in the Modeler.
- **Show Dataflow View:** Launches the Dataflow view.
- **Simulate:** Tests the Schema Map policy. For more information, see Section 6.3, “Testing Schema Mapping Policies,” on page 79.
- **Export to 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 Schema Map policy.
- **Properties:** Allows you to rename the Schema Map policy.

6.5.2 Policy Flow View Additional Options

1 Right-click the Schema Map policy in the Policy Flow view.



- **Add Policy > DirXML Script:** Adds a new Schema Map policy using DirXML® Script.
- **Add Policy > XSLT:** Adds a new Schema Map policy using XSLT.
- **Add Policy > Schema Mapping:** 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 Mapping:** Launches the Schema Map editor. For more information, see Section 6.2, "Editing a Schema Mapping Policy," on page 77.
- **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.

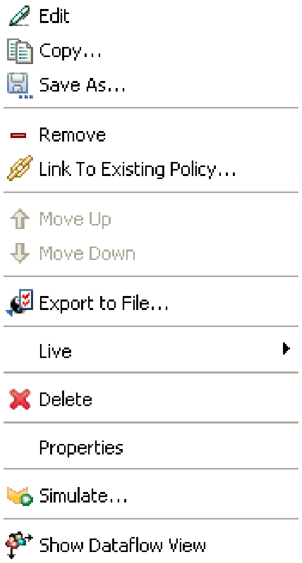


- **Live > Import Driver:** Imports an existing driver from the Identity Vault.
- **Live > Deploy Driver:** Deploys the existing driver into the Identity Vault.

- **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.
- **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 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.
- **Simulate:** Tests the Schema Map policy. For more information, see Section 6.3, "Testing Schema Mapping Policies," on page 79.

6.5.3 Policy Set View Additional Options

1 Right-click the Schema Map policy in the Policy Set view.



- **Edit:** Launches the Schema Map editor. For more information, see Section 6.2, "Editing a Schema Mapping Policy," on page 77.
- **Copy:** Creates a copy of the Schema Map policy.
- **Save As:** Saves the Schema Map policy as a `.xml` file.
- **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 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 Schema Map policy.
- **Properties:** Allows you to rename the Schema Map policy.
- **Simulate:** Tests the Schema Map policy. For more information, see Section 6.3, "Testing Schema Mapping Policies," on page 79.
- **Show Dataflow View:** Launches the Dataflow view.

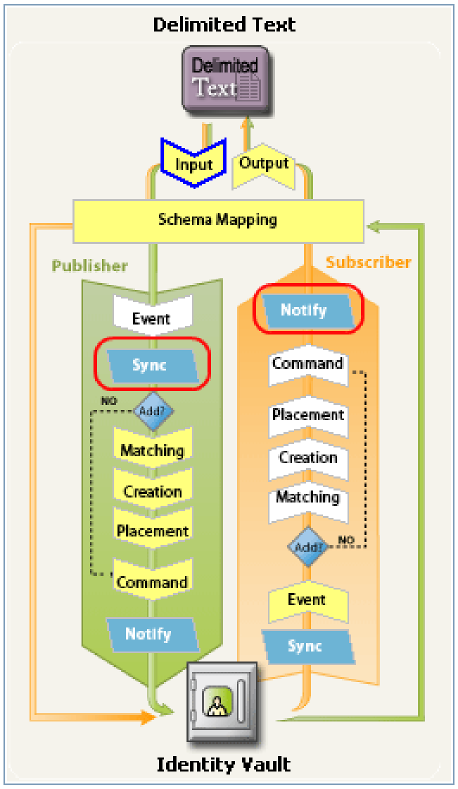
7 Controlling the Flow of Objects with the Filter

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.

- Section 7.1, "Accessing the Filter Editor," on page 88
- Section 7.2, "Editing the Filter," on page 91
- Section 7.3, "Testing the Filter," on page 96
- Section 7.4, "Viewing the Filter in XML," on page 99
- Section 7.5, "Additional Filter Options," on page 99

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.

Figure 7-1 *Filter in Policy Flow View*



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 Metadirectory engine, but the object is not automatically synchronized. For more information, see Section 7.2.5, "Changing the Filter Settings," on page 93.

7.1 Accessing the Filter Editor

The Filter editor allows you to edit the filter. There are three different ways to access the Filter editor: through the model outline, through the policy flow, and through the Policy Set view.

- Section 7.1.1, “Model Outline View,” on page 88
- Section 7.1.2, “Policy Flow View,” on page 88
- Section 7.1.3, “Policy Set View,” on page 90
- Section 7.1.4, “Keyboard Support,” on page 90

7.1.1 Model Outline View

1 In an open project, click the *Outline* tab.

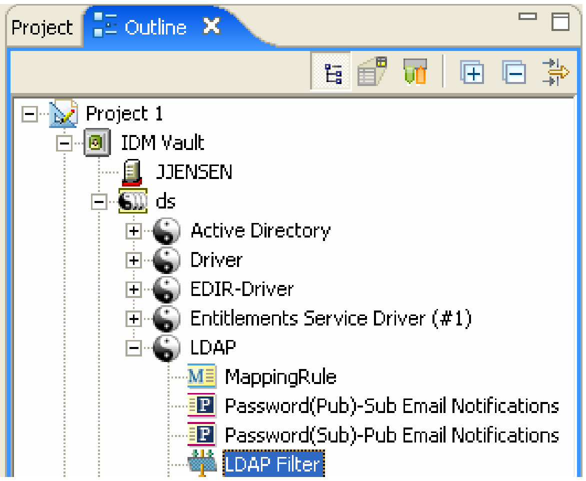
2 Click the *Show Model Outline* icon.

3 Select the driver you want to manage the filter for, then click the plus sign to the right.

4 Double-click the *Filter* icon and to launch the Filter editor.

or

Right-click and select *Edit*.

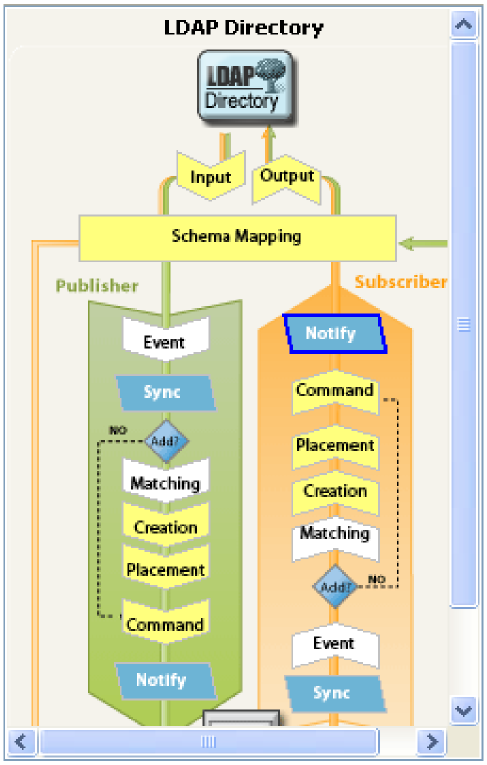

Project
Outline
Project 1
IDM Vault
JJENSEN
ds
Active Directory
Driver
EDIR-Driver
Entitlements Service Driver (#1)
LDAP
MappingRule
Password(Pub)-Sub Email Notifications
Password(Sub)-Pub Email Notifications
LDAP Filter

7.1.2 Policy Flow View

1 In an open project, click the *Outline* tab.

2 Select the *Show Policy Flow* icon.

3 Double-click the *Sync* icon or the *Notify* icon to launch the Filter editor.

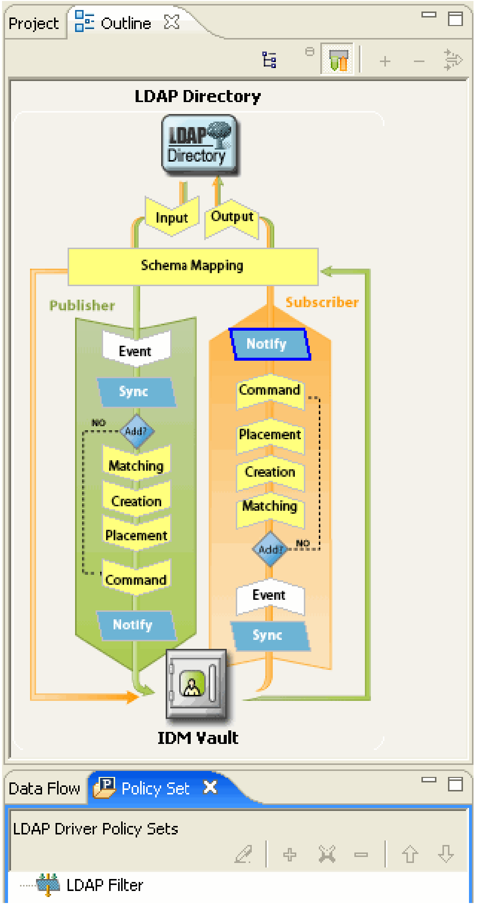


or

Right-click and select *Edit Policy* > *Filter*.

7.1.3 Policy Set View

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



7.1.4 Keyboard Support

***Table 7-1** Filter Editor Keyboard Support*

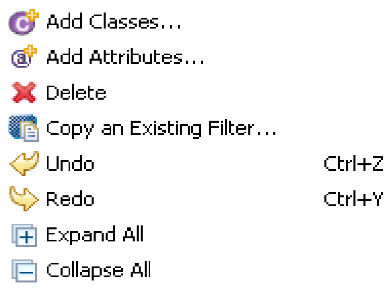
Action	Description
Up-arrow	Moves the cursor up in the Filter editor.
Down-arrow	Moves the cursor down in the Filter editor.
Left-arrow	Collapses the information displayed
Right-arrow	Expands the information displayed.
Insert	Adds a class.
Ctrl+Insert	Adds an attribute.
Delete	Deletes the selected items.
Enter	Accesses the edit mode. Press Enter a second time to save the changes.

Action	Description
Esc	Exits the edit mode.

7.2 Editing the Filter

The Filter editor allows you to create and edit the filter. To display a context menu, right-click an item.

Figure 7-2 *Filter Options*



- Section 7.2.1, “Removing or Adding Classes and Attributes,” on page 91
- Section 7.2.2, “Modifying Multiple Attributes,” on page 92
- Section 7.2.3, “Copying an Existing Filter,” on page 92
- Section 7.2.4, “Setting Default Values for Attributes,” on page 92
- Section 7.2.5, “Changing the Filter Settings,” on page 93

7.2.1 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

If you do not want a class or an attribute to synchronize, the best practice is to completely remove the class or the attribute completely from the filter. There are two different ways to add or remove attributes and classes from the filter:

- Right-click the class or attribute you want to remove, then select *Delete*.
- Select the class or attribute you want to remove, then click the *Delete* icon ✖ in the upper right corner.

Adding a Class

1. Right-click in the Filter editor, then click *Add Classes*.

   or

   Click the *Add Classes* icon in the upper right corner
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 Right-click in the Filter editor, then click *Add Attribute*.

or

Click the *Attribute* icon in the upper right corner.

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*.

7.2.2 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.

7.2.3 Copying an Existing Filter

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

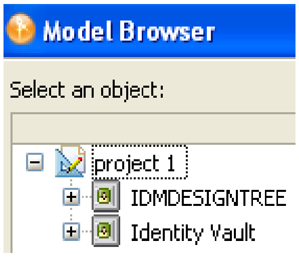
1 Click the *Copy an Existing Filter* icon

or

Right-click in the Filter editor, then click *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.



7.2.4 Setting Default Values for Attributes

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

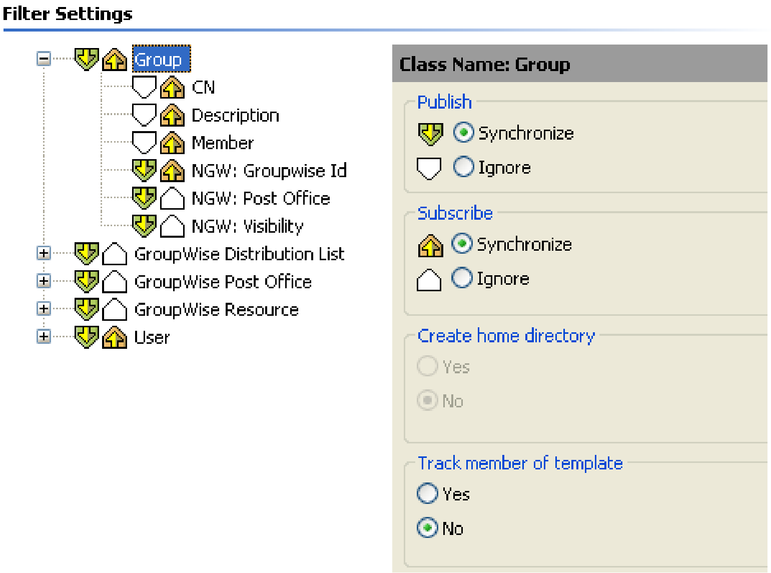
1 Click the *Set Default Values for New Attributes* icon in the upper right corner.

2 Select the options you want new attributes to have, then click *OK*.

7.2.5 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.



Filter Settings

- Group
  - CN
  - Description
  - Member
  - NGW: Groupwise Id
  - NGW: Post Office
  - NGW: Visibility
- GroupWise Distribution List
- GroupWise Post Office
- GroupWise Resource
- User

Class Name: Group

Publish: ● Synchronize ○ Ignore

Subscribe: ● Synchronize ○ Ignore

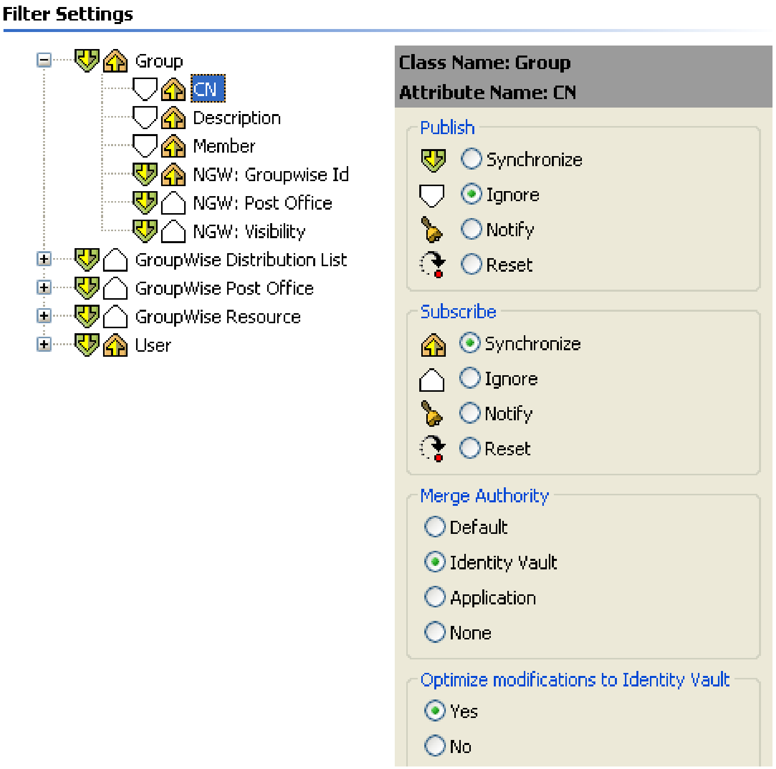
Create home directory: ○ Yes ● No

Track member of template: ○ Yes ● No

2 Change the filter settings for the selected class.

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.
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 eDirectory. The option only works for 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 eDirectory Template object, the eDirectory driver maintains the Member of Template attribute, if the *Track Member of Template* option is selected. The option only works for eDirectory.

3 Select an attribute.



Filter Settings

- Group
  - CN
  - Description
  - Member
  - NGW: Groupwise Id
  - NGW: Post Office
  - NGW: Visibility
- GroupWise Distribution List
- GroupWise Post Office
- GroupWise Resource
- User

Class Name: Group
Attribute Name: CN

Publish

○ Synchronize
● Ignore
○ Notify
○ Reset

Subscribe

● Synchronize
○ Ignore
○ Notify
○ Reset

Merge Authority

○ Default
● Identity Vault
○ Application
○ None

Optimize modifications to Identity Vault

● Yes
○ No

4 Change the filter settings for the selected attribute.

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.
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.

Options	Definitions
Optimize Modification to Identity Manager	◆ **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 Metadirectory 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.

5 Click the *Save* icon to save the changes.

7.3 Testing the Filter

Designer comes with a tool called the Policy Simulator, which allows you to test your 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 the *Launch Policy Simulator* icon in the toolbar.

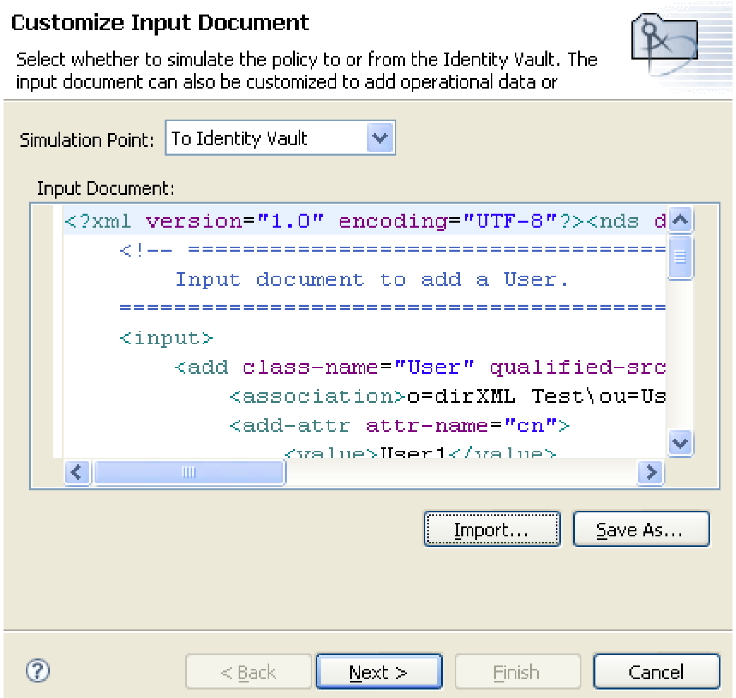
2 Select *To Identity Vault* or *From Identity Vault* as the simulation point of the filter.

3 Select *Import* to browse to a file that simulates an event.

4 Select the file, then click *Open*. This example uses the `com.novell.designer.idm.policy\simulation\add\User.xml` file, which simulates an Add event for a User object.

The Policy Simulator displays the input document of the user Add event.

5 Click *Next* to begin the simulation.



Customize Input Document

Select whether to simulate the policy to or from the Identity Vault. The input document can also be customized to add operational data or

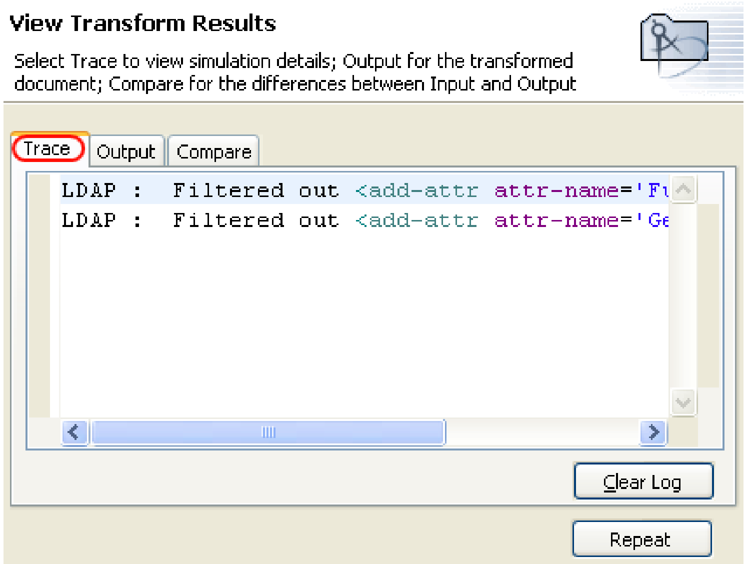
Simulation Point: To Identity Vault

Input Document:

```
<?xml version="1.0" encoding="UTF-8"?><nds d
    <!-- ===================================
        Input document to add a User.
    ========================================
    <input>
        <add class-name="User" qualified-src
            <association>o=dirXML Test\ou=Us
            <add-attr attr-name="cn">
                <value>User1</value>
```

The Policy Simulator displays the log of the Add event, the output document, and a comparison of the Input document to the Output document that is generated.

6 Select the *Trace* tab to display the results of the Add event as you would through DSTRACE.



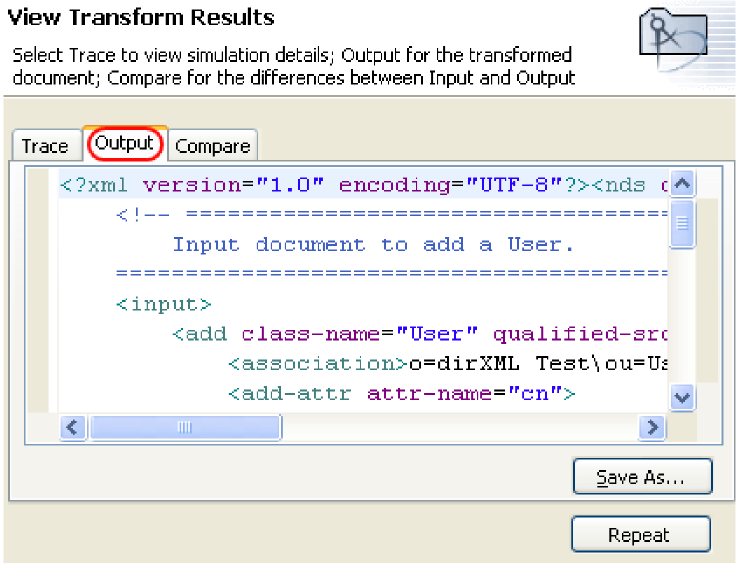
View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output

```
LDAP :  Filtered out <add-attr attr-name='Fı
LDAP :  Filtered out <add-attr attr-name='Ge
```

Click *Clear Log*, then click *Repeat* to run the simulation again with new trace log.

7 Select the *Output* tab to see the output document that is generated when the filter is executed against an input document. The input document is the user Add event.



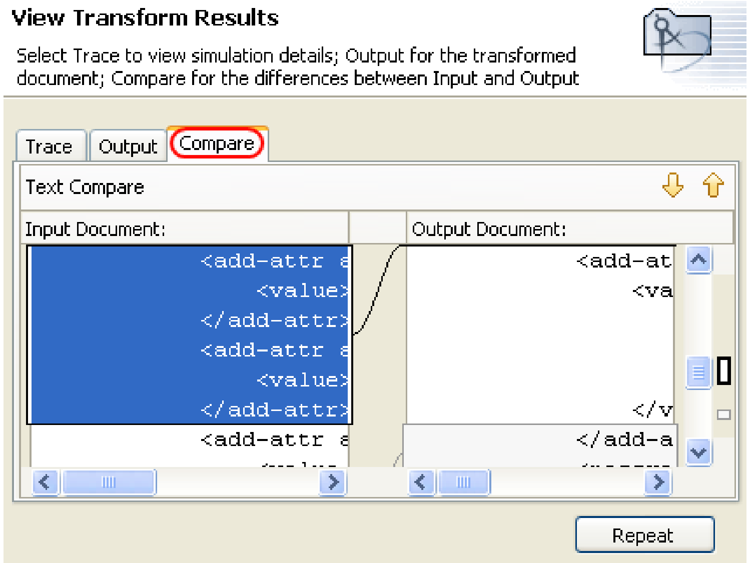
View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output

```
<?xml version="1.0" encoding="UTF-8"?><nds d
    <!-- ===================================
        Input document to add a User.
    ========================================
    <input>
        <add class-name="User" qualified-src
            <association>o=dirXML Test\ou=Us
            <add-attr attr-name="cn">
```

You can edit the input and output documents. If you want to keep the changes, click *Save As*.

8 Select the *Compare* tab to compare the text of the input document to the output document that is generated.



View Transform Results

Select Trace to view simulation details; Output for the transformed document; Compare for the differences between Input and Output

```
<add-attr a
    <value>
</add-attr>
<add-attr a
    <value>
</add-attr>
<add-attr a
```

```
<add-at
    <va


    </v
</add-a
```

9 Click *Repeat* to select a different input document and see the results of that event.

10 When you have finished testing the filter, click *Finish* to close the Policy Simulator.

7.4 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 Novell XML Editor” in the *Designer 2.1 for Identity Manager 3.5.1*.

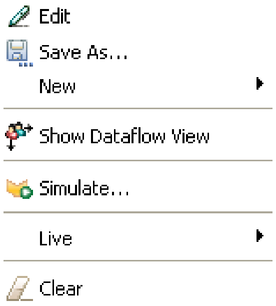
7.5 Additional Filter Options

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.

- Section 7.5.1, “Outline View Additional Options,” on page 99
- Section 7.5.2, “Policy Flow View Additional Options,” on page 100
- Section 7.5.3, “Policy Set View Additional Options,” on page 100

7.5.1 Outline View Additional Options

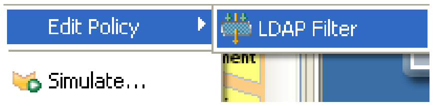
1 Right-click the filter object in the Outline view.



- **Edit:** Launches the Filter editor. For more information, see Section 7.2, “Editing the Filter,” on page 91.
- **Save As:** Saves the filter as a `.xml` file.
- **New:** Creates a Domain Group or an Identity Vault in the modeler.
- **Show Dataflow View:** Launches the Dataflow view.
- **Simulate:** Launches the Policy Simulator. For more information, see Section 7.3, “Testing the Filter,” on page 96.
- **Live > Deploy:** Deploys the filter into the Identity Vault.
- **Clear:** Deletes all content from the filter policy, but leaves the object.

7.5.2 Policy Flow View Additional Options

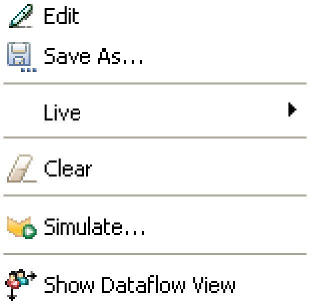
1. Right-click the filter object in the Policy Flow view.



- **Edit Policy > Filter:** Launches the Filter editor. For more information, see Section 7.2, "Editing the Filter," on page 91.
- **Simulate:** Launches the Policy Simulator. For more information, see Section 7.3, "Testing the Filter," on page 96.

7.5.3 Policy Set View Additional Options

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



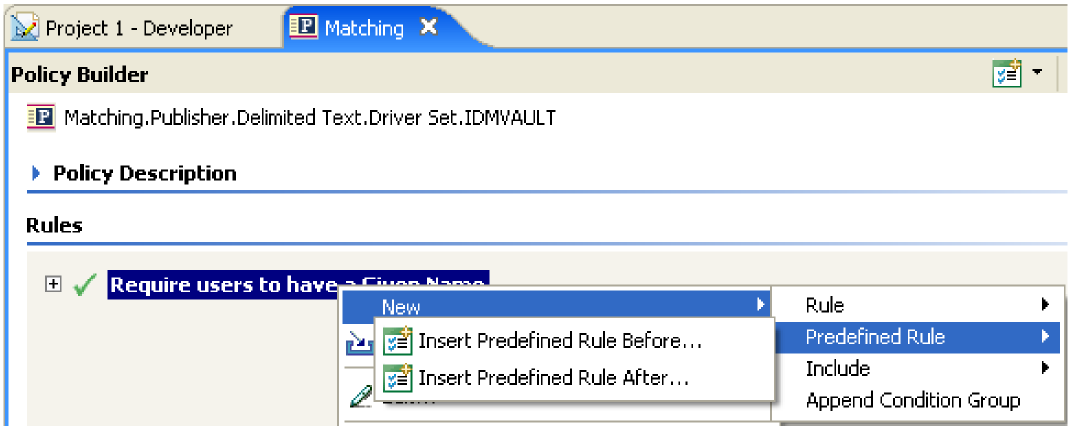
- **Edit:** Launches the Filter editor. For more information, see Section 7.2, "Editing the Filter," on page 91.
- **Save As:** Saves the filter as a XML file.
- **Live > Deploy:** Allows you to deploy the filter into the Identity Vault.
- **Clear:** Deletes all content from the filter policy, but leaves the object.
- **Simulate:** Launches the Policy Simulator. For more information, see Section 7.3, "Testing the Filter," on page 96.
- **Show Dataflow View:** Launches the Dataflow view.

8 Using Predefined Rules

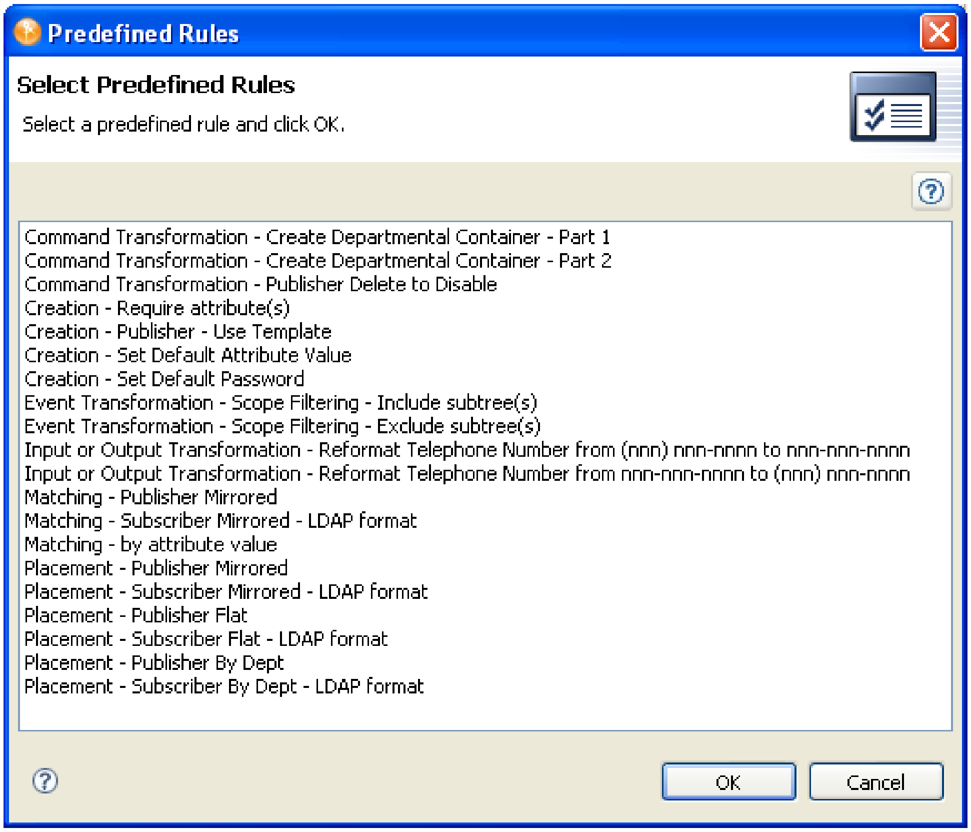
Designer includes 19 predefined rules. You can import and use these rules as well as create your own rules. These rules include common tasks that administrators use. You need to provide information specific to your environment to customize the rules.

To access the predefined rules:

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


Project 1 - Developer
Matching
Policy Builder
Matching.Publisher.Delimited Text.Driver Set.IDMVAULT
Policy Description
Rules
Require users to have a Given Name
New
Insert Predefined Rule Before...
Insert Predefined Rule After...
Rule
Predefined Rule
Include
Append Condition Group

The Predefined Rules dialog box displays a list of the available rules.



Select a predefined rule and click OK.

Command Transformation - Create Departmental Container - Part 1
Command Transformation - Create Departmental Container - Part 2
Command Transformation - Publisher Delete to Disable
Creation - Require attribute(s)
Creation - Publisher - Use Template
Creation - Set Default Attribute Value
Creation - Set Default Password
Event Transformation - Scope Filtering - Include subtree(s)
Event Transformation - Scope Filtering - Exclude subtree(s)
Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn
Input or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn
Matching - Publisher Mirrored
Matching - Subscriber Mirrored - LDAP format
Matching - by attribute value
Placement - Publisher Mirrored
Placement - Subscriber Mirrored - LDAP format
Placement - Publisher Flat
Placement - Subscriber Flat - LDAP format
Placement - Publisher By Dept
Placement - Subscriber By Dept - LDAP format

8.1 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” on page 103.

8.1.1 Creating a Policy

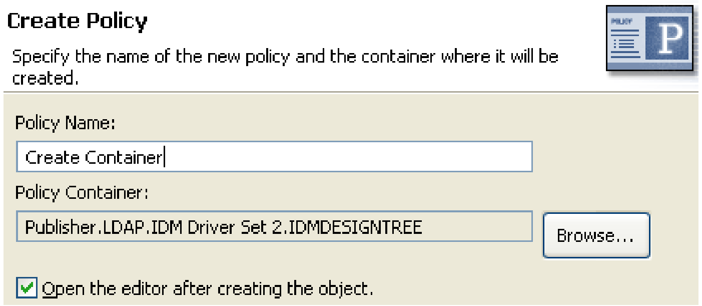
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Create Container

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

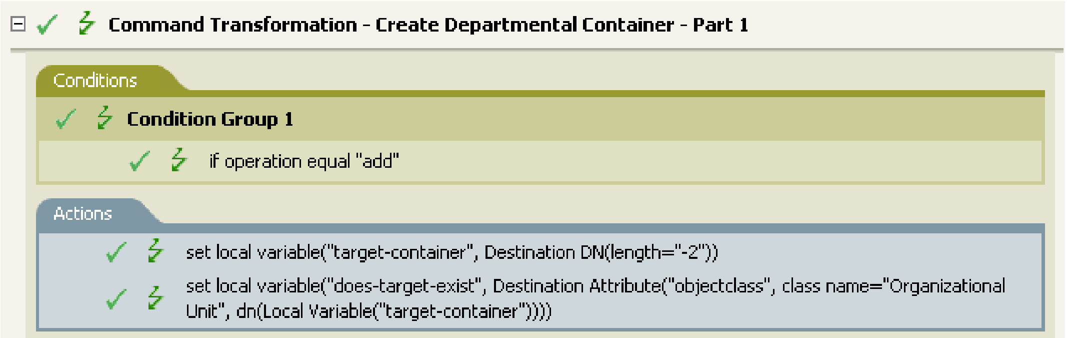
8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor’s changes and continue?`” Click *Yes*. The Policy Builder is launched and the new Command Transformation policy is saved.

9 Continue with Section 8.1.2, “Importing the Predefined Rule,” on page 103.

8.1.2 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*.



Command Transformation - Create Departmental Container - Part 1

Conditions

Condition Group 1

if operation equal "add"

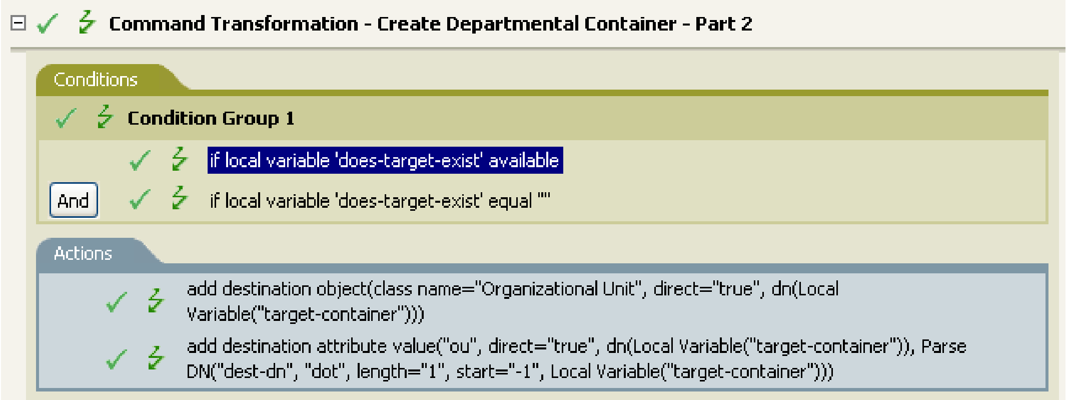
Actions

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"))))

3 Right-click in 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*.



Command Transformation - Create Departmental Container - Part 2

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")))

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

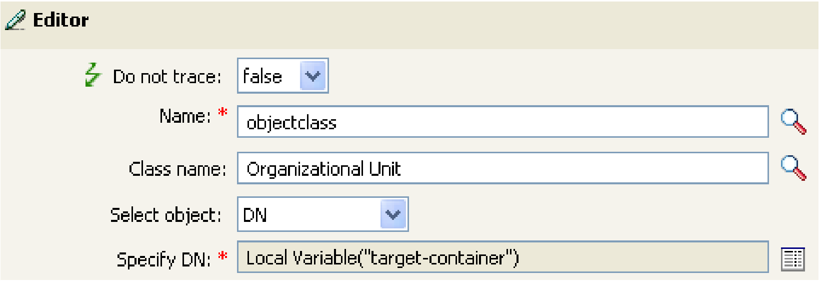
There is no information to change that is specific to your environment.

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

8.1.3 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 OrganizationalUnit is set to the local variable of target-container.



Do not trace: false

Name: * objectclass

Class name: Organizational Unit

Select object: DN

Specify DN: * Local Variable("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.

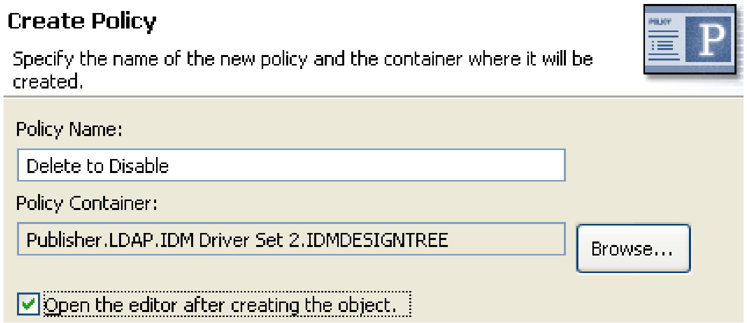
8.2 Command Transformation - Publisher Delete to Disable

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" on page 106.

8.2.1 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. 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.



Specify the name of the new policy and the container where it will be created.

Policy Name: Delete to Disable

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

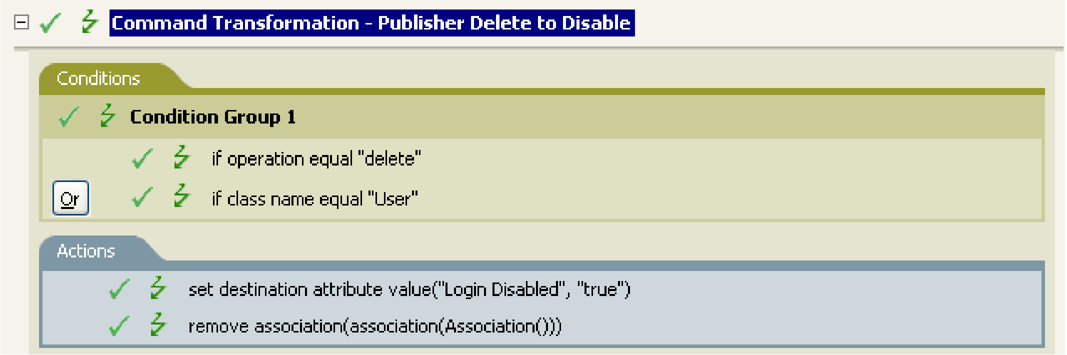
8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor’s changes and continue?`” Click *Yes*. The Policy Builder is launched and the new Command Transformation policy is saved.

9 Continue with Section 8.2.2, “Importing the Predefined Rule,” on page 106.

8.2.2 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*.



Command Transformation - Publisher Delete to Disable

Condition Group 1

if operation equal "delete"

if class name equal "User"

set destination attribute value("Login Disabled", "true")

remove association(association(Association()))

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

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

8.2.3 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 into the Novell® eDirectory™ tree, but the User object was not deleted.

8.3 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” on page 107.

8.3.1 Creating a Policy

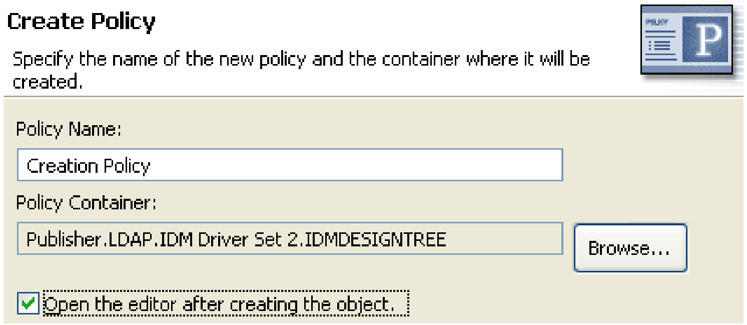
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Creation Policy

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

Browse...

☑ Open the editor after creating the object.

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

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

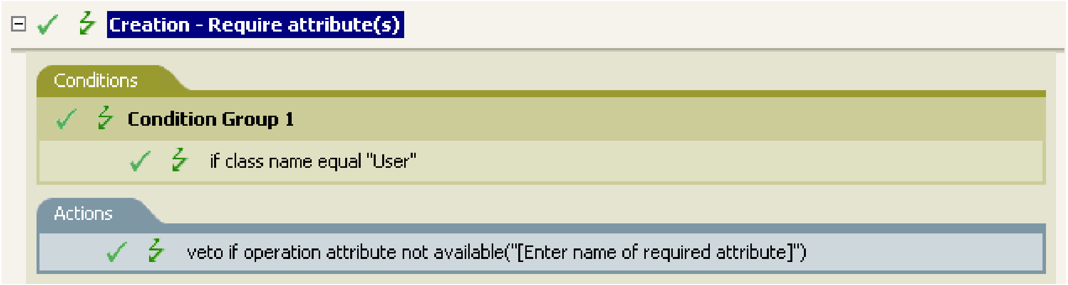
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Creation policy is saved.

9 Continue with Section 8.3.2, "Importing the Predefined Rule," on page 107.

8.3.2 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*.



Creation - Require attribute(s)

Conditions

Condition Group 1

if class name equal "User"

Actions

veto if operation attribute not available("[Enter name of required attribute]")

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

4 Delete *[Enter name of required attribute]* from the *Enter Name* field.

5 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*.

8.3.3 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 action and select *New > Append Action*. Select *veto if operation attribute not available*, then browse to the attribute you want to require.

8.4 Creation - Publisher - Use Template

This rule allows the use of a Novell 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" on page 109.

8.4.1 Creating a Policy

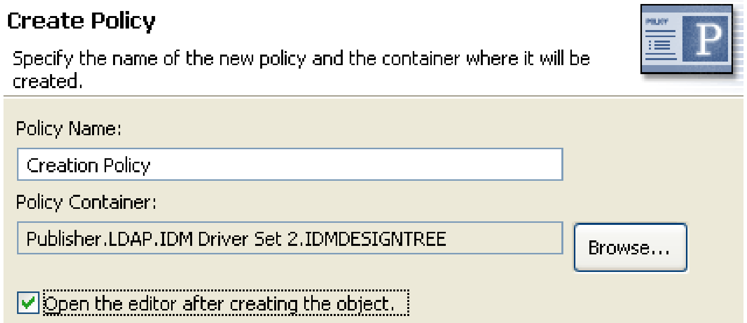
1 From the *Outline* view or the *Policy Flow* view, select the Publisher channel.

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 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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Creation Policy

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

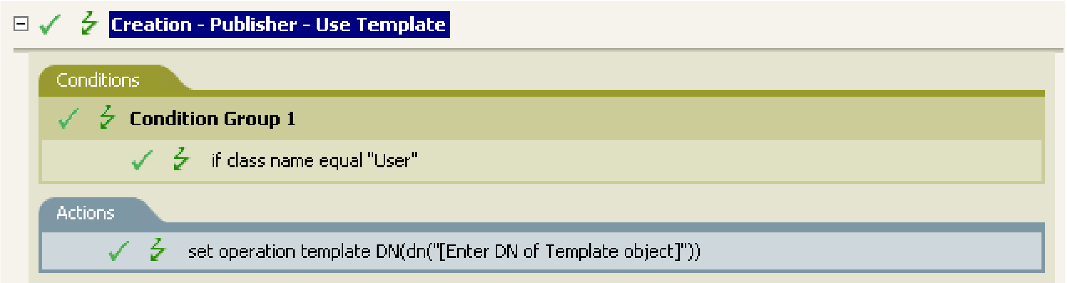
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Creation policy is saved.

9 Continue with Section 8.4.2, "Importing the Predefined Rule," on page 109.

8.4.2 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*.



Creation - Publisher - Use Template

Conditions

Condition Group 1

if class name equal "User"

Actions

set operation template DN(dn("[Enter DN of Template object]"))

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

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

5 Click the *Edit Arguments* icon ▤ to launch the Argument Builder.

6 Select *Text* in the *Noun* list.

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

8 In the Editor, click the browse icon, browse to and select the template object, then click *OK*.

9 Click *OK*.

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

8.4.3 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, 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.

8.5 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" on page 110.

8.5.1 Creating a Policy

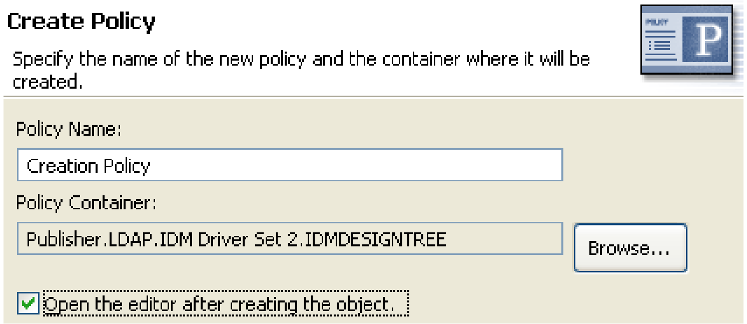
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Creation Policy

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

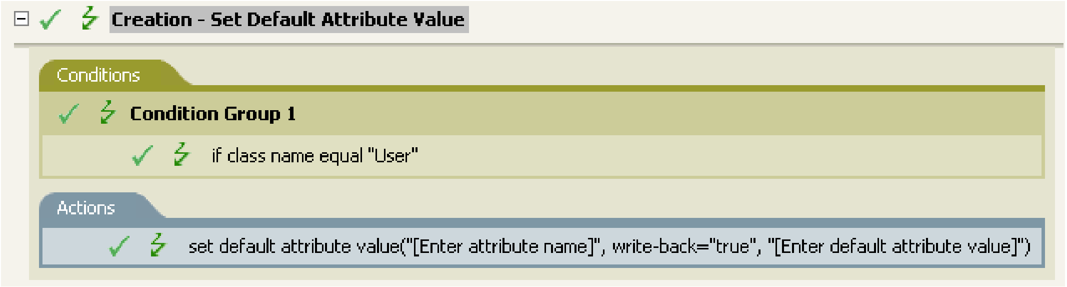
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Creation policy is saved.

9 Continue with Section 8.5.2, "Importing the Predefined Rule," on page 110.

8.5.2 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*.



Creation - Set Default Attribute Value

Condition Group 1

if class name equal "User"

set default attribute value("[Enter attribute name]", write-back="true", "[Enter default attribute value]")

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

4 Delete *[Enter attribute name]* from the *Enter attribute name* field.

5 Click the browse icon, then browse to and select the attribute you want to create.

6 Delete *[Enter default attribute value]* from the *Enter arguments values* field.

7 Click the *Edit Arguments* icon ▤ to launch the Argument Values List Builder.

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

9 Click the *Edit Arguments* icon ▤ to launch the Argument Builder.

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

11 Click *OK*.

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

8.5.3 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 action and click *New* > *Append Action*. Select the action, set the default attribute value, and follow Step 1 on page 110 through Step 12 on page 111 to assign the value to the attribute.

8.6 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" on page 112.

8.6.1 Creating a Policy

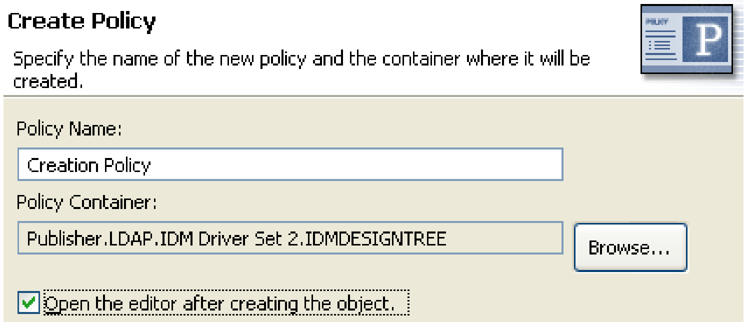
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Creation Policy

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

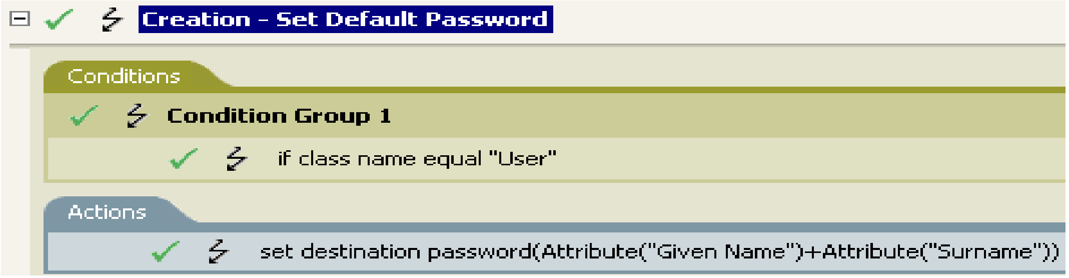
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Creation policy is saved.

9 Continue with Section 8.6.2, "Importing the Predefined Rule," on page 112.

8.6.2 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*.



Creation - Set Default Password

Conditions

Condition Group 1

if class name equal "User"

Actions

set destination password(Attribute("Given Name")+Attribute("Surname"))

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

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

8.6.3 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 set the password to any other value you want through the Argument Builder.

8.7 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 (page 113).

8.7.1 Creating a Policy

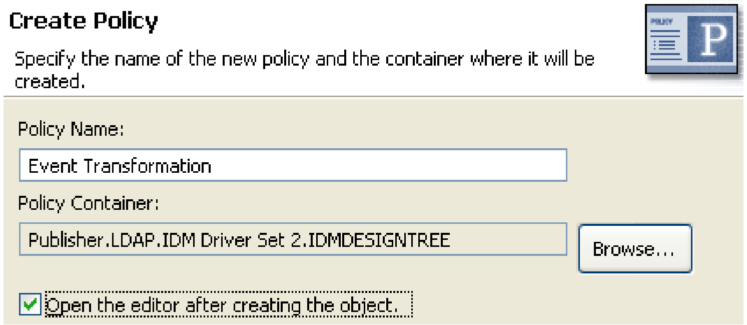
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Event Transformation

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

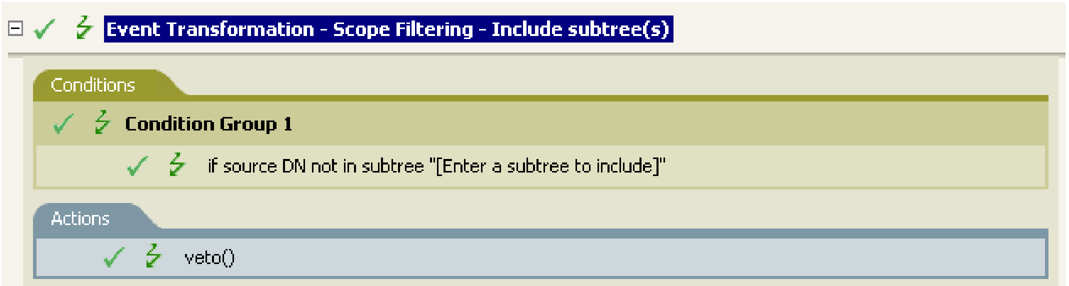
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Event Transformation policy is saved.

9 Continue with Section 8.7.2, "Importing the Predefined Rule," on page 113.

8.7.2 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*.



Event Transformation - Scope Filtering - Include subtree(s)

Condition Group 1

if source DN not in subtree "[Enter a subtree to include]"

veto()

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

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

5 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*.

6 Click *OK*.

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

8.7.3 How the Rule Works

This rule is used when you want to exclude part of the Identity Vault from synchronizing. It allows you 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.

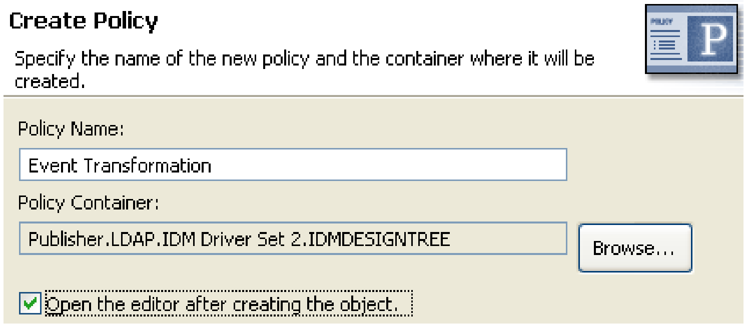
8.8 Event Transformation - Scope Filtering - Exclude Subtrees

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" on page 115.

8.8.1 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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Event Transformation

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

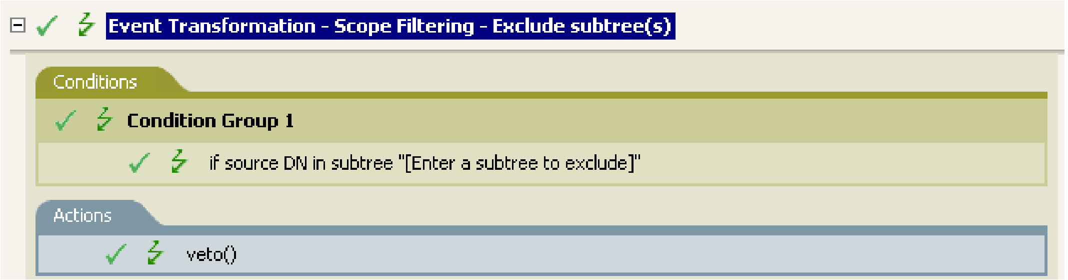
☑ Open the editor after creating the object.

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 Section 8.8.2, "Importing the Predefined Rule," on page 115.

8.8.2 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*.



Event Transformation - Scope Filtering - Exclude subtree(s)

Condition Group 1

if source DN in subtree "[Enter a subtree to exclude]"

veto()

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

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

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

6 Click *OK*.

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

8.8.3 How the Rule Works

This rule is used when you want to exclude part of the Identity Vault from synchronizing. It allows you 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.

8.9 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" on page 116.

8.9.1 Creating a Policy

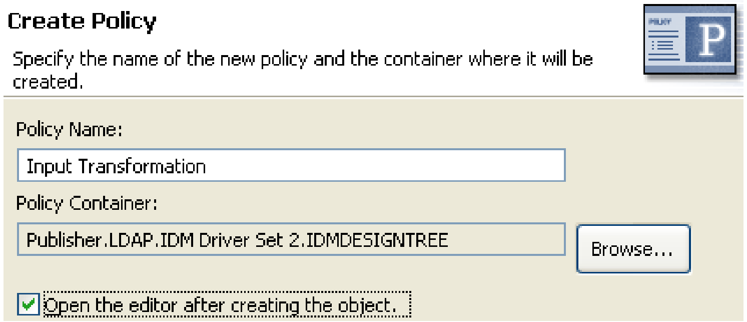
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Input Transformation

Policy Container:

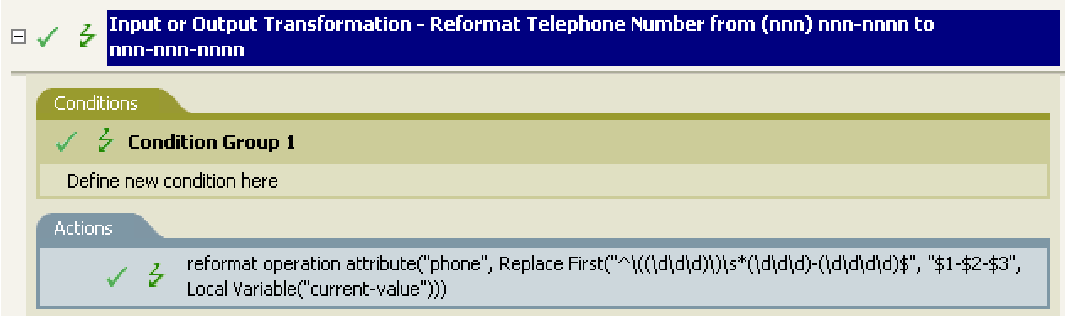
Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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*. Policy Builder is launched and the new Input or Output Transformation policy is saved.
9. Continue with Section 8.9.2, “Importing the Predefined Rule,” on page 116.

8.9.2 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*.



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

Condition Group 1

Define new condition here

```
reformat operation attribute("phone", Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3",
Local Variable("current-value")))
```

3. Edit the condition by double-clicking the *Conditions* tab.
4. Define the condition you want to have occur when the telephone number is reformatted.
5. Click *OK*.
6. Save the rule by clicking *File* > *Save*.

8.9.3 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.

8.10 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" on page 117.

8.10.1 Creating a Policy

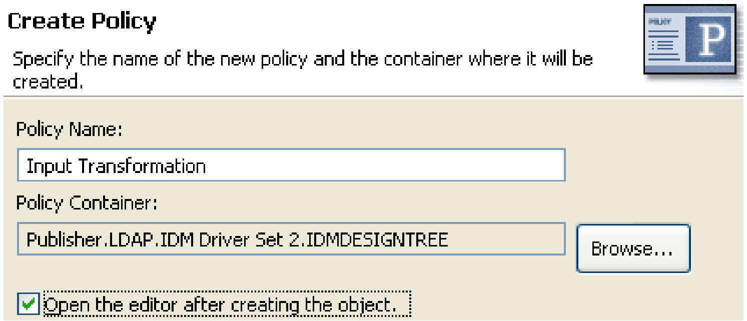
1 From the *Outline* view or the *Policy Flow* view, select the Publisher or Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Input Transformation

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

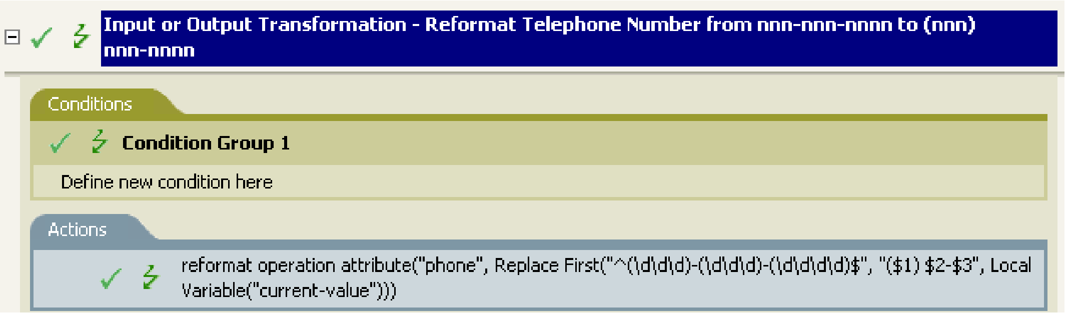
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. Policy Builder is launched and the new Input or Output Transformation policy is saved.

9 Continue with Section 8.10.2, "Importing the Predefined Rule," on page 117.

8.10.2 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 I*nput or Output Transformation - Reformat Telephone Number from nnn-nnn-nnnn to (nnn) nnn-nnnn*, then click *OK*.



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

Condition Group 1

Define new condition here

```
reformat operation attribute("phone", Replace First("^(\d\d\d)-(\d\d\d)-(\d\d\d\d)$", "($1) $2-$3", Local Variable("current-value")))
```

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

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

5 Click *OK*.

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

8.10.3 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.

8.11 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 (page 119).

8.11.1 Creating a Policy

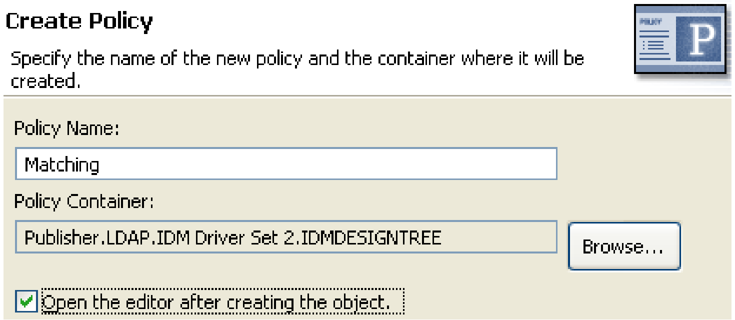
1 From the *Outline* view or the *Policy Flow* view, select the Publisher channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Matching

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

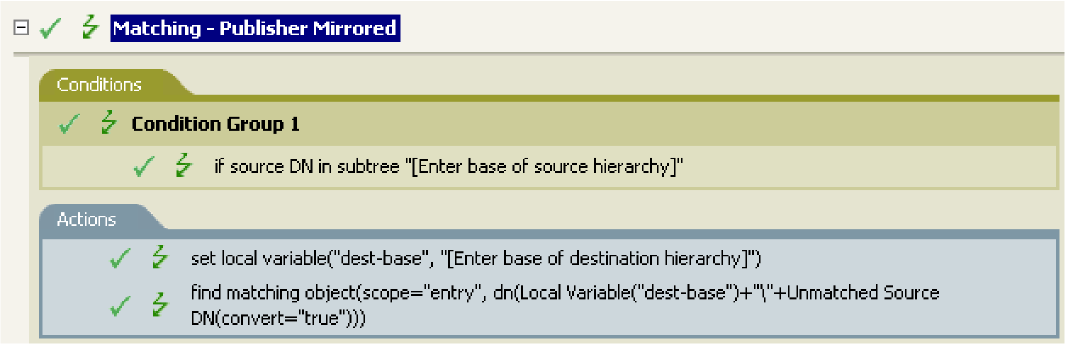
8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor’s changes and continue?`” Click *Yes*. The Policy Builder is launched and the new Matching policy is saved.

9 Continue with Section 8.11.2, “Importing the Predefined Rule,” on page 119.

8.11.2 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*.



Matching - Publisher Mirrored

Condition Group 1

if source DN in subtree "[Enter base of source hierarchy]"

set local variable("dest-base", "[Enter base of destination hierarchy]")

find matching object(scope="entry", dn(Local Variable("dest-base")+"\"+Unmatched Source DN(convert="true")))

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

4 Delete *[Enter base of source hierarchy]* from the *Value* field.

5 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 *Enter string* field.

9 Click the *Edit 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 *OK*.

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

8.11.3 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.

8.12 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" on page 121.

8.12.1 Creating a Policy

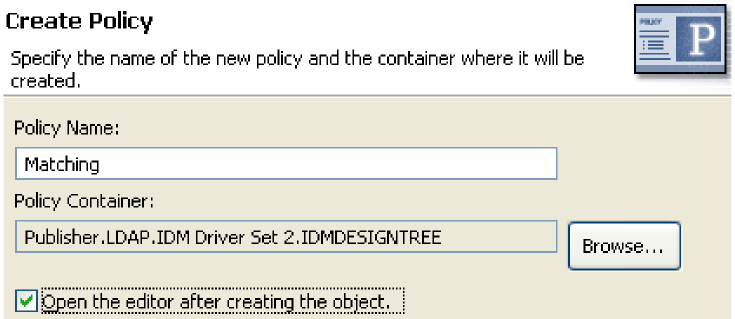
1 From the *Outline* view or the *Policy Flow* view, select the Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Matching

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

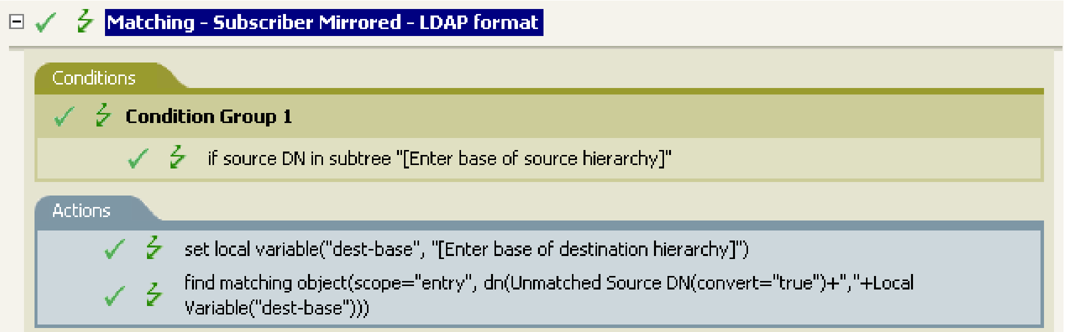
6 Select *Open Editor after creating policy*, then click *Next*.

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 Section 8.12.2, “Importing the Predefined Rule,” on page 121.

8.12.2 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*.



⊟ ✓ Matching - Subscriber Mirrored - LDAP format

Conditions

✓ Condition Group 1

✓ if source DN in subtree "[Enter base of source hierarchy]"

Actions

✓ set local variable("dest-base", "[Enter base of destination hierarchy]")

✓ find matching object(scope="entry", dn(Unmatched Source DN(convert="true")+","+Local Variable("dest-base")))

3. Edit the condition by double-clicking the *Conditions* tab.
4. Delete *[Enter base of source hierarchy]* from the *Value* field.
5. 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 *Enter String* field.
9. Click the *Edit 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 *OK*.
14. Save the rule by clicking *File* > *Save*.

8.12.3 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.

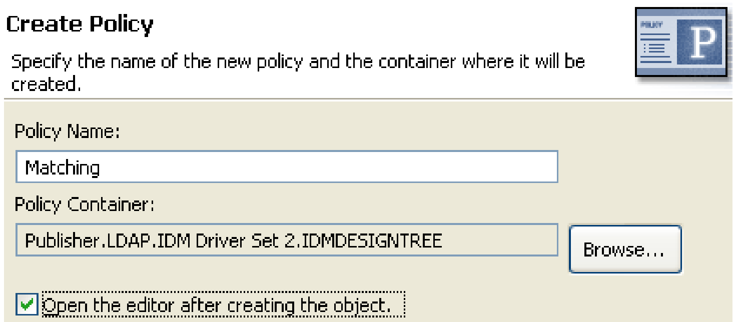
8.13 Matching - By Attribute Value

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" on page 122.

8.13.1 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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Matching

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

Browse...

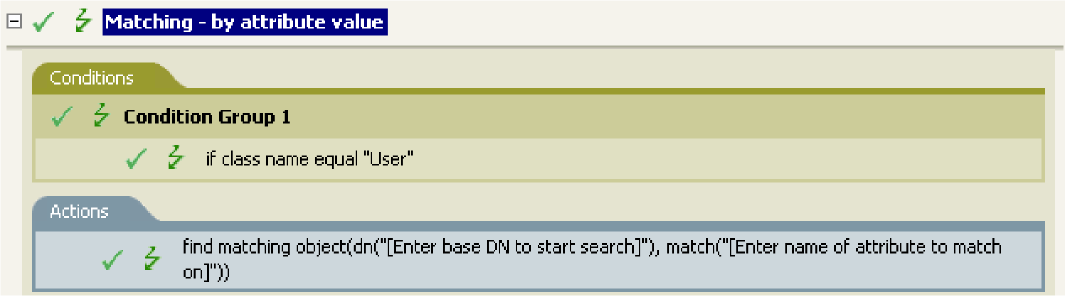
☑ Open the editor after creating the object.

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 Section 8.13.2, "Importing the Predefined Rule," on page 122.

8.13.2 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*.



⊟ ✓ Matching - by attribute value

Conditions

✓ Condition Group 1

✓ if class name equal "User"

Actions

✓ find matching object(dn("[Enter base DN to start search]"), match("[Enter name of attribute to match on]"))

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

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

5 Click the *Edit Arguments* icon ▤ to launch the Argument Builder.

6 Select *Text* in the *Noun* list.

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

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

9 Delete *[Enter name of attribute to match on]* from the *Enter Match Attributes* field.

10 Click the *Edit Arguments* icon ▤ to launch the Match Attributes Builder.

11 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*.

12 Click *OK*.

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

8.13.3 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 they attributes do not exist, a new User object is created.

8.14 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" on page 124.

8.14.1 Creating a Policy

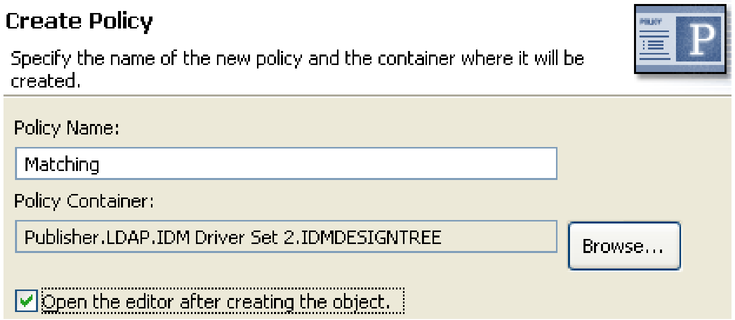
1 From the *Outline* view or the *Policy Flow* view, select the Publisher channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Matching

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

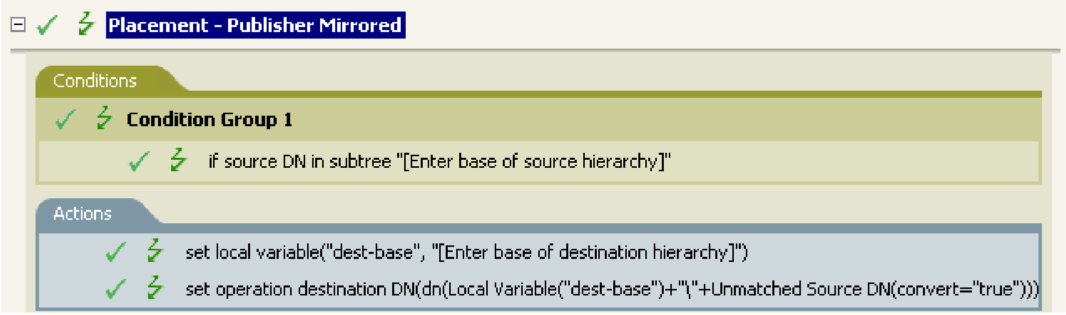
8 A file conflict window appears with the message “`Before editing this item you need to save. Do you wish to save the editor’s changes and continue?`” Click *Yes*. The Policy Builder is launched and the new Placement policy is saved.

9 Continue with Section 8.14.2, “Importing the Predefined Rule,” on page 124.

8.14.2 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*.



Placement - Publisher Mirrored

Conditions

Condition Group 1

if source DN in subtree "[Enter base of source hierarchy]"

Actions

set local variable("dest-base", "[Enter base of destination hierarchy]")

set operation destination DN(dn(Local Variable("dest-base")+"\"+Unmatched Source DN(convert="true")))

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

4 Delete *[Enter base of source hierarchy]* from the *Value* field.

5 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 *Enter String* field.

8 Click the *Edit 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 *OK*.

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

8.14.3 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.

8.15 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" on page 126.

8.15.1 Creating a Policy

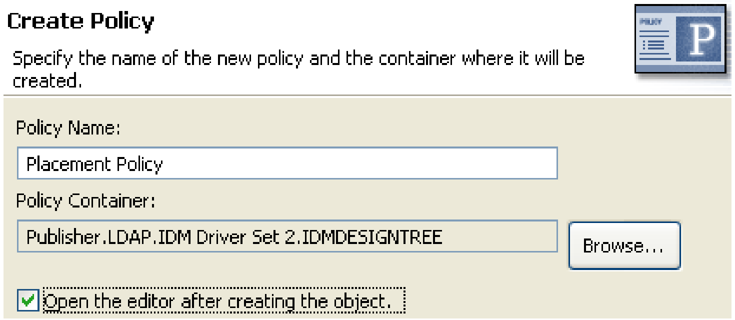
1 From the *Outline* view or the *Policy Flow* view, select the Subscriber channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Placement Policy

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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`

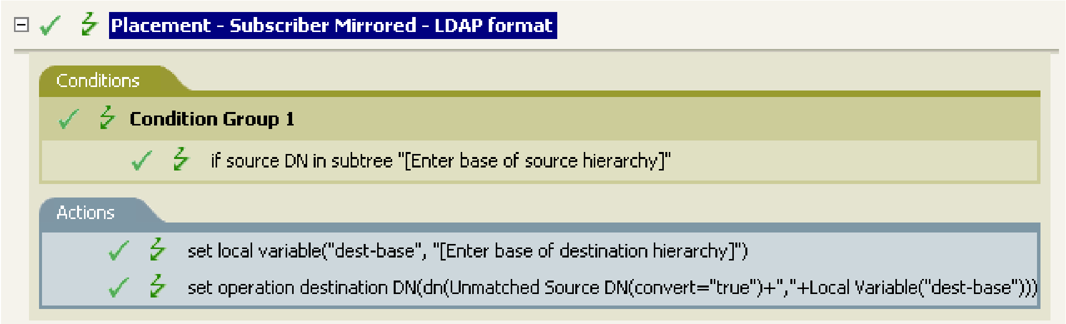
`continue?`” Click *Yes*. The Policy Builder is launched and the new Placement policy is saved.

9 Continue with Section 8.15.2, “Importing the Predefined Rule,” on page 126.

8.15.2 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*.



Placement - Subscriber Mirrored - LDAP format

Conditions

Condition Group 1

if source DN in subtree "[Enter base of source hierarchy]"

Actions

set local variable("dest-base", "[Enter base of destination hierarchy]")

set operation destination DN(dn(Unmatched Source DN(convert="true")+","+Local Variable("dest-base")))

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

4 Delete *[Enter base of source hierarchy]* from the *Value* field.

5 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 *Enter String* field.

8 Click the *Edit 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 *OK*.

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

8.15.3 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 unmatched source DN, dest-base. The rule uses LDAP format.

8.16 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" on page 127.

8.16.1 Creating a Policy

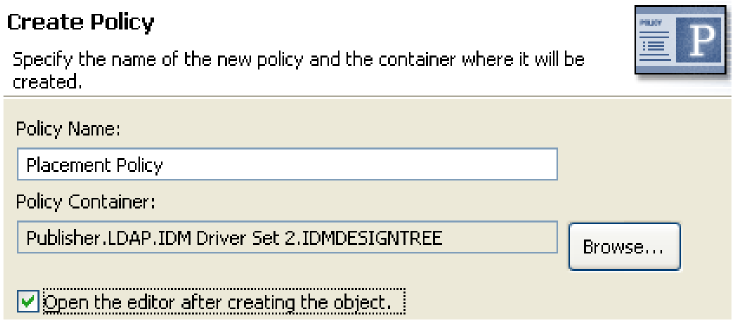
1 From the *Outline* view or the *Policy Flow* view, select the Publisher channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Placement Policy

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

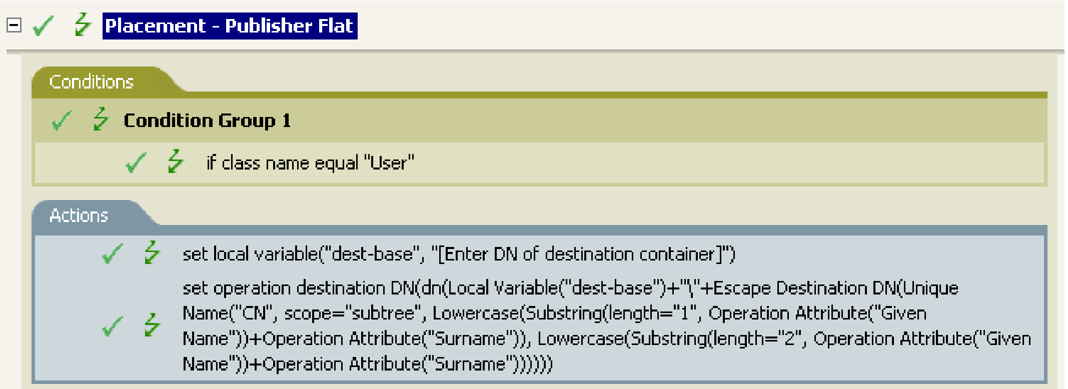
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Placement policy is saved.

9 Continue with Section 8.16.2, "Importing the Predefined Rule," on page 127.

8.16.2 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*.



```
if class name equal "User"
```

```
set local variable("dest-base", "[Enter DN of destination container]")
set operation destination DN(dn(Local Variable("dest-base")+"\"+Escape Destination DN(Unique
Name("CN", scope="subtree", Lowercase(Substring(length="1", Operation Attribute("Given
Name"))+Operation Attribute("Surname")), Lowercase(Substring(length="2", Operation Attribute("Given
Name"))+Operation Attribute("Surname"))))))
```

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

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

5 Click the *Edit Arguments* icon ▤ to launch the Argument Builder.

6 Select *Text* in the *Noun* list.

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

8 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*.

9 Click *OK*.

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

8.16.3 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 as lowercase. The rule uses slash format.

8.17 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" on page 129.

8.17.1 Creating a Policy

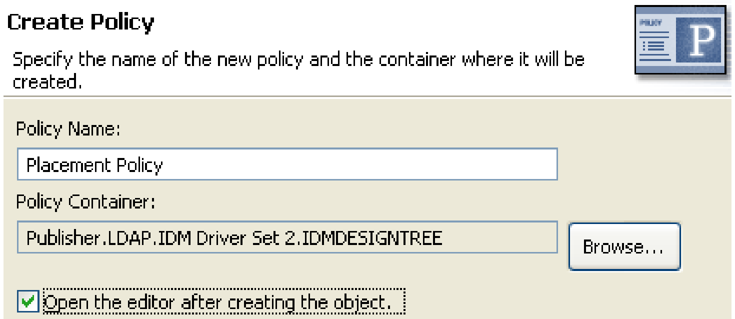
1 From the *Outline* view or the *Policy Flow* view, select the Publisher channel.

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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Placement Policy

Policy Container:

Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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

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

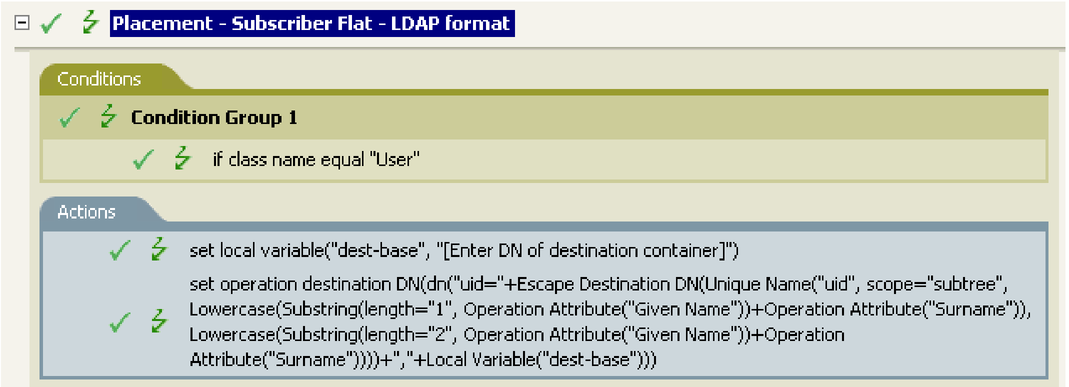
8 A file conflict window appears with the message "`Before editing this item you need to save. Do you wish to save the editor's changes and continue?`" Click *Yes*. The Policy Builder is launched and the new Placement policy is saved.

9 Continue with Section 8.17.2, "Importing the Predefined Rule," on page 129.

8.17.2 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*.



Placement - Subscriber Flat - LDAP format

Conditions

Condition Group 1

if class name equal "User"

Actions

set local variable("dest-base", "[Enter DN of destination container]")

set operation destination DN(dn("uid="+Escape Destination DN(Unique Name("uid", scope="subtree", Lowercase(Substring(length="1", Operation Attribute("Given Name"))+Operation Attribute("Surname")), Lowercase(Substring(length="2", Operation Attribute("Given Name"))+Operation Attribute("Surname"))))+","+Local Variable("dest-base")))

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

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

5 Click the *Edit Arguments* icon to launch the Argument Builder.

6 Select *Text* in the *Noun* list.

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

8 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*.

9. Click *OK*.
10. Save the rule by clicking *File* > *Save*.

8.17.3 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.

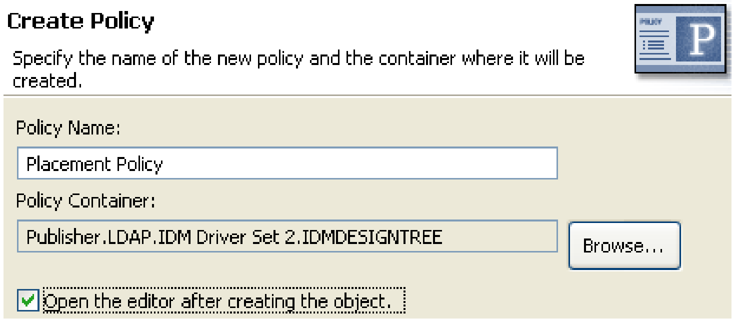
8.18 Placement - Publisher By Dept

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" on page 131.

8.18.1 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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name:

Placement Policy

Policy Container:

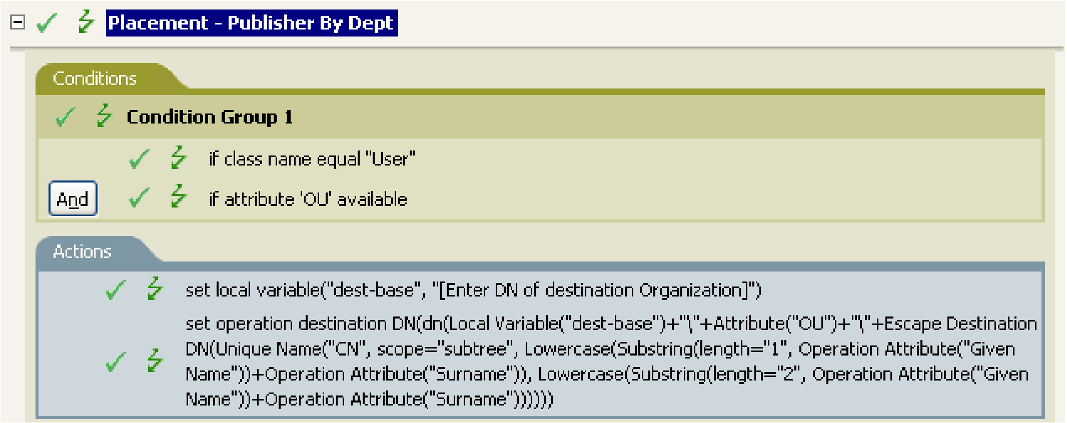
Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

☑ Open the editor after creating the object.

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 Section 8.18.2, "Importing the Predefined Rule," on page 131.

8.18.2 Importing the Predefined Rule

1. Right-click in 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*.



Placement - Publisher By Dept

Conditions

Condition Group 1

if class name equal "User"

And if attribute 'OU' available

Actions

set local variable("dest-base", "[Enter DN of destination Organization]")

set operation destination DN(dn(Local Variable("dest-base")+"\"+Attribute("OU")+"\"+Escape Destination DN(Unique Name("CN", scope="subtree", Lowercase(Substring(length="1", Operation Attribute("Given Name"))+Operation Attribute("Surname")), Lowercase(Substring(length="2", Operation Attribute("Given Name"))+Operation Attribute("Surname"))))))

3. Edit the action by double-clicking the *Actions* tab.
4. Delete *[Enter DN of destination Organization]* from the *Enter String* fields.
5. Click the *Edit Arguments* icon ▤ to launch the Argument Builder.
6. Select *Text* in the *Noun* list.
7. Double-click *Text* to add it to the argument.
8. 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*.
9. Click *OK*.
10. Save the rule by clicking *File* > *Save*.

8.18.3 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.

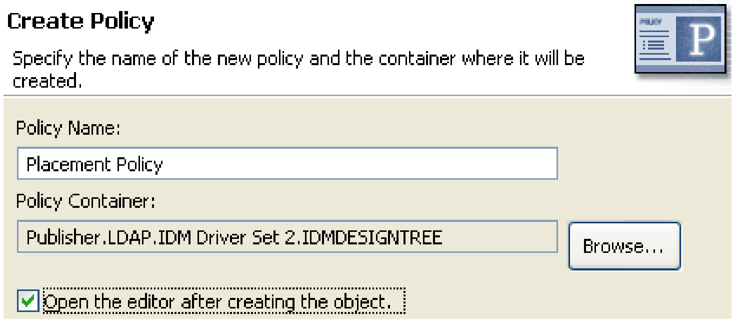
8.19 Placement - Subscriber By Dept - LDAP Format

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" on page 132.

8.19.1 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.



Create Policy

Specify the name of the new policy and the container where it will be created.

Policy Name: Placement Policy

Policy Container: Publisher.LDAP.IDM Driver Set 2.IDMDESIGNTREE

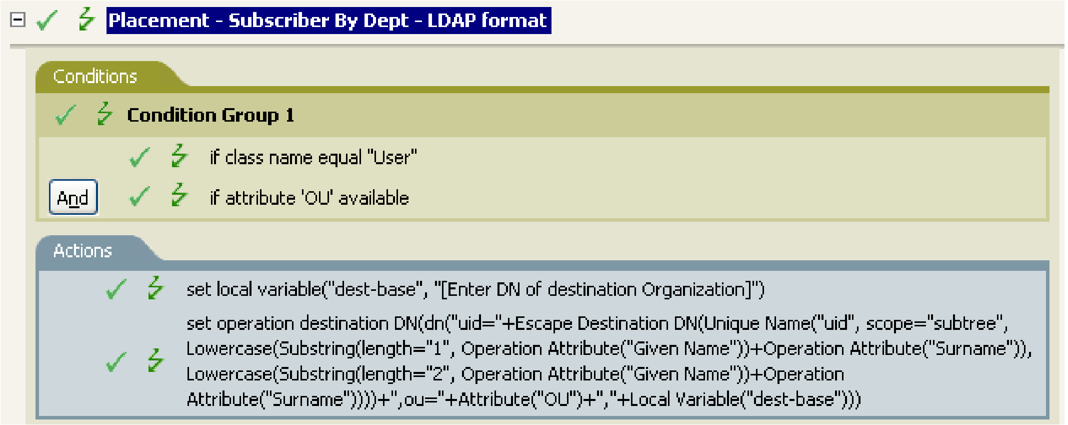
☑ Open the editor after creating the object.

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 Section 8.19.2, "Importing the Predefined Rule," on page 132.

8.19.2 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*.



Placement - Subscriber By Dept - LDAP format

Conditions

Condition Group 1

```
if class name equal "User"
And  if attribute 'OU' available
```

Actions

```
set local variable("dest-base", "[Enter DN of destination Organization]")
set operation destination DN(dn("uid="+Escape Destination DN(Unique Name("uid", scope="subtree",
Lowercase(Substring(length="1", Operation Attribute("Given Name"))+Operation Attribute("Surname")),
Lowercase(Substring(length="2", Operation Attribute("Given Name"))+Operation
Attribute("Surname"))))+",ou="+Attribute("OU")+","+Local Variable("dest-base")))
```

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

4 Delete *[Enter DN of destination Organization]* from the *Enter string* field.

5 Click the *Edit Arguments* icon ▤ to launch the Argument Builder.

6 Select *Text* in the *Noun* list.

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

8 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*.

9 Click *OK*.

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

8.19.3 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.

Testing Policies with the Policy Simulator

9

The Policy Simulator allows you to execute a policy at any point in the flow of the driver and see the results without implementing the policy in the Identity Vault. 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:

- Section 9.1, "Accessing the Policy Simulator," on page 135
- Section 9.2, "Using the Policy Simulator," on page 137
- Section 9.3, "Simulating Policies with Java Extensions," on page 140

The Policy Simulator uses XML. The eDirectory™ document type definition file (`nds.dtd`) defines the schema of the XML documents that the Metadirectory 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 find information about why errors are occurring, see the "NDS DTD" in the *Identity Manager 3.5.1 DTD Reference*.

If the policy uses a mapping table object or ECMAScript object, the Policy Simulator tests these object when the policy is tested.

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.

9.1 Accessing the Policy Simulator

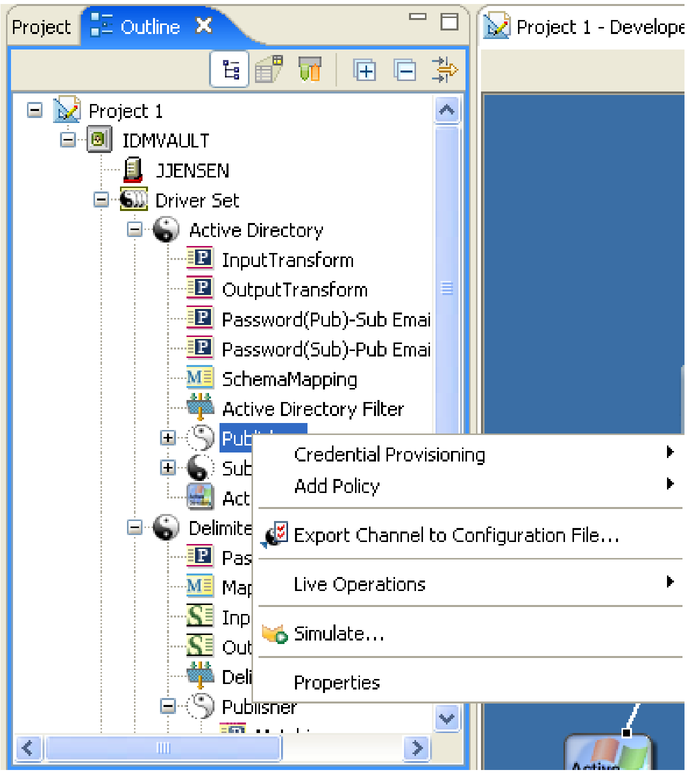
The Policy Simulator can be accessed in three different ways:

- "Outline View" on page 135
- "Policy Flow View" on page 136
- "Editors" on page 137

9.1.1 Outline View

1 Click the *Show Model Outline* icon.

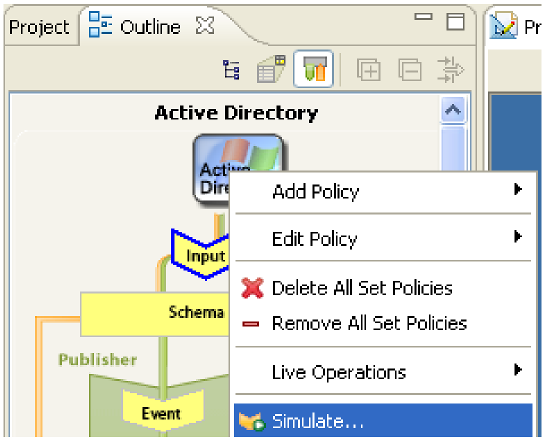
2 Right-click the driver, publisher, subscriber, mapping rule, filter, or any policy you want to simulate, then click *Simulate*.



9.1.2 Policy Flow View

1 Click the *Show Policy Flow* icon .

2 Right-click the Input, Output, Schema Mapping, filter, or any policy set icons you want to simulate, then click *Simulate*.



9.1.3 Editors

You can access the Policy Simulator through the Policy Builder, the Schema Mapping editor, or the Filter editor by selecting the *Policy Simulator* icon in the toolbar of each editor.

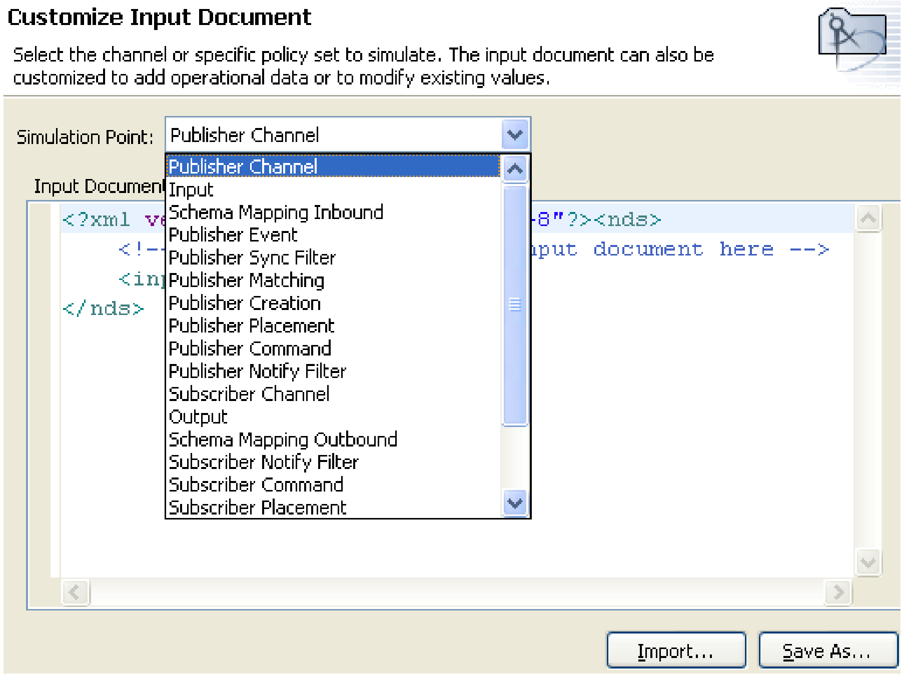
9.2 Using the Policy Simulator

The Policy Simulator allows you to select a point in the driver flow to test the policy with a specific operation. It allows you to edit the input and output documents while you are testing. If you want to keep the changes, select the *Save As* icon to save the document as an XML file.

To use the Policy Simulator:

1 From the Simulation Point drop-down list, select the place in the driver flow that you want to test the policy.

You can select the any of the following items: Publisher Channel, Subscriber Channel, Input, Schema Mapping, Event, Sync Filter, Matching, Creation, Placement, Command and Notify Filter.


Customize Input Document

Select the channel or specific policy set to simulate. The input document can also be customized to add operational data or to modify existing values.

Simulation Point: Publisher Channel

Input Document

```
<?xml ve-8"?><nds>
    <!--nput document here -->
    <in
</nds>
```

If you select a specific policy or rule to test, the Simulation Point option only shows *To Identity Vault* or *From Identity Vault*.

2 Select *Import*, then browse to and select a file to test.

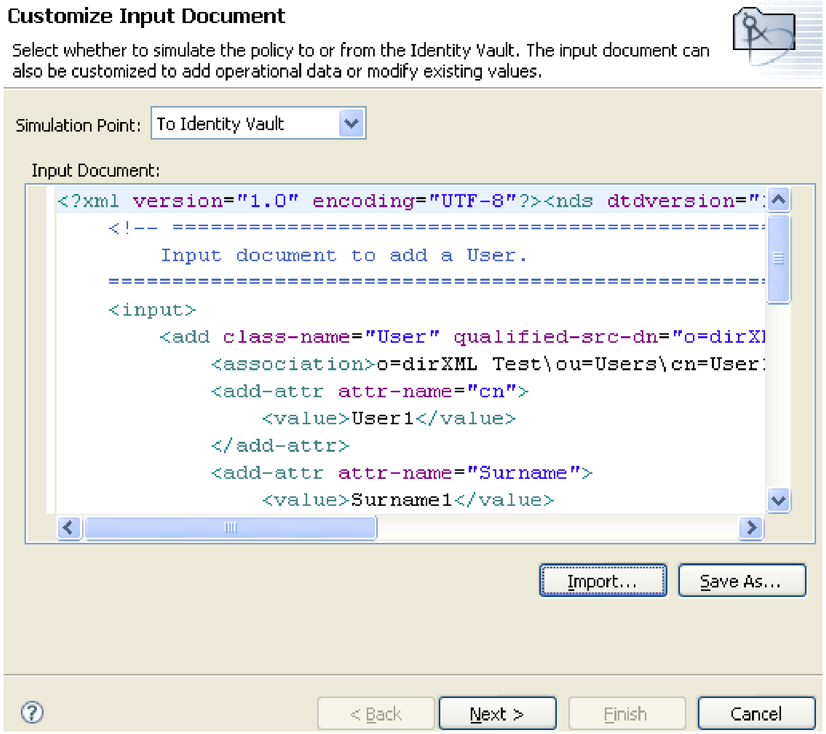
Designer comes with sample event files you can use. The files are located in the plug-in `com.novell.designer.idm.policy\simulation.` The event are Add, Association, Delete, Instance, Modify, Move, Query, Rename, and Status.

3 Double-click a folder and 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* to display the input document in the window.

5 Click *Next* to begin the simulation.



Customize Input Document

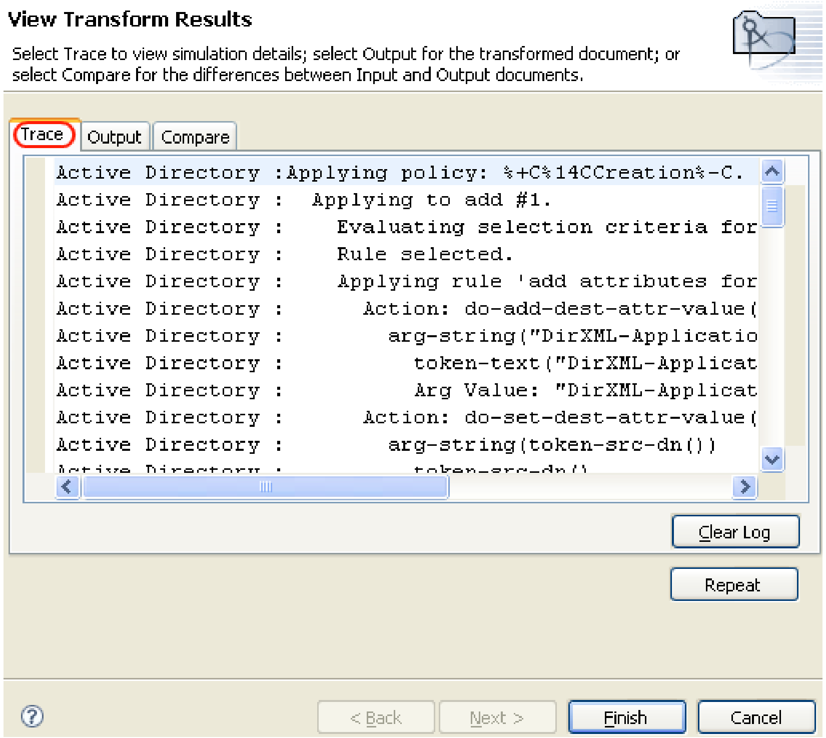
Select whether to simulate the policy to or from the Identity Vault. The input document can also be customized to add operational data or modify existing values.

Simulation Point: To Identity Vault

Input Document:

```
<?xml version="1.0" encoding="UTF-8"?><nds dtdversion="
    <!-- ===================================================
        Input document to add a User.
    ======================================================
    <input>
        <add class-name="User" qualified-src-dn="o=dirX
            <association>o=dirXML Test\ou=Users\cn=User
            <add-attr attr-name="cn">
                <value>User1</value>
            </add-attr>
            <add-attr attr-name="Surname">
                <value>Surname1</value>
```

6 Select the *Trace* tab to display the results of the Add event as you would through DSTRACE.



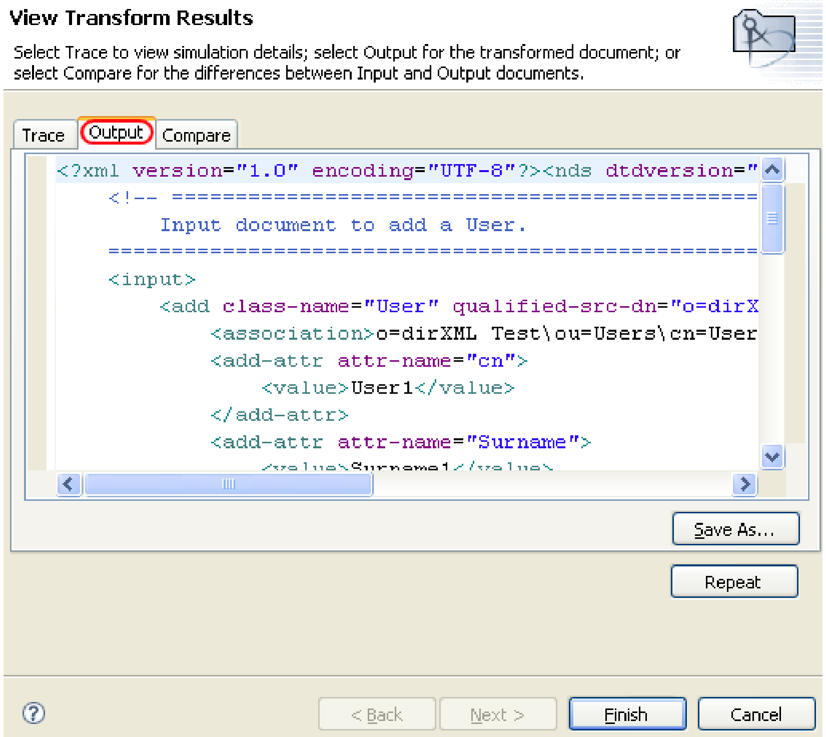
View Transform Results

Select Trace to view simulation details; select Output for the transformed document; or select Compare for the differences between Input and Output documents.

```
Active Directory :Applying policy: %+C%14CCreation%-C.
Active Directory :  Applying to add #1.
Active Directory :    Evaluating selection criteria for
Active Directory :    Rule selected.
Active Directory :    Applying rule 'add attributes for
Active Directory :      Action: do-add-dest-attr-value(
Active Directory :        arg-string("DirXML-Applicatio
Active Directory :          token-text("DirXML-Applicat
Active Directory :          Arg Value: "DirXML-Applicat
Active Directory :      Action: do-set-dest-attr-value(
Active Directory :        arg-string(token-src-dn())
Active Directory :          token-src-dn()
```

Click *Clear Log*, then click *Repeat* to run the simulation again with the new trace log.

7 Select the *Output* tab to see the output document that is generated when the policy is executed against an input document. The input document is the user Add event.



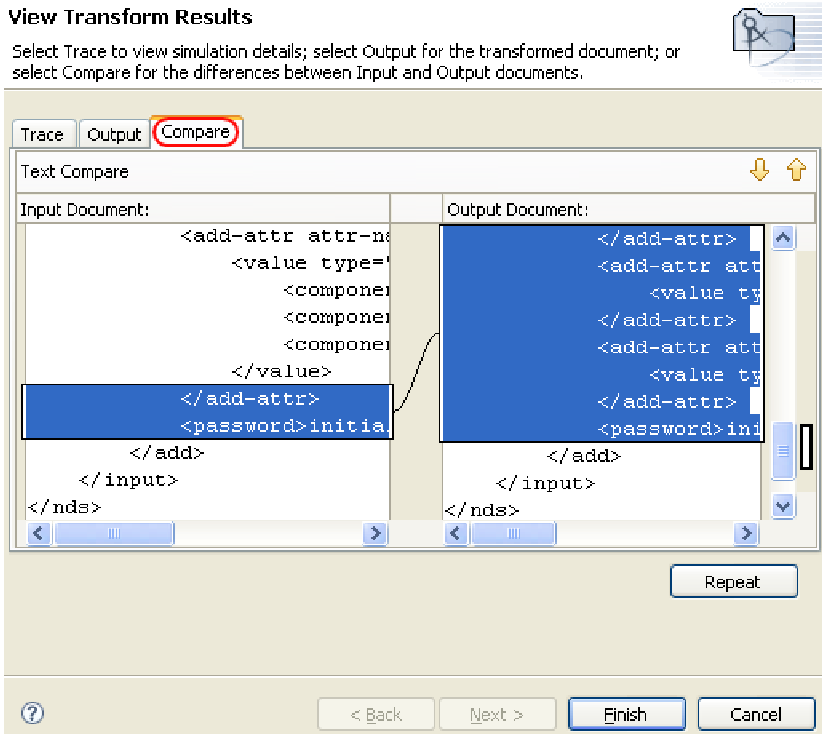
View Transform Results

Select Trace to view simulation details; select Output for the transformed document; or select Compare for the differences between Input and Output documents.

```
<?xml version="1.0" encoding="UTF-8"?><nds dtdversion="
    <!-- =============================================
        Input document to add a User.
    ===================================================
    <input>
        <add class-name="User" qualified-src-dn="o=dirX
            <association>o=dirXML Test\ou=Users\cn=User
            <add-attr attr-name="cn">
                <value>User1</value>
            </add-attr>
            <add-attr attr-name="Surname">
                <value>Surname1</value>
```

You can edit the input and output documents. If you want to keep the changes, click *Save As*.

8 Select the *Compare* tab to compare the output document to the input document.



View Transform Results

Select Trace to view simulation details; select Output for the transformed document; or select Compare for the differences between Input and Output documents.

Input Document:

```
            <add-attr attr-na
                <value type='
                    <componei
                    <componei
                    <componei
                </value>
            </add-attr>
            <password>initia
        </add>
    </input>
</nds>
```

Output Document:

```
            </add-attr>
            <add-attr att
                <value ty
            </add-attr>
            <add-attr att
                <value ty
            </add-attr>
            <password>ini
        </add>
    </input>
</nds>
```

9 Click *Repeat* to select a different input document and see the results of that event.

10 When you are finished testing, click *Finish* to close the Policy Simulator.

9.3 Simulating Policies with Java Extensions

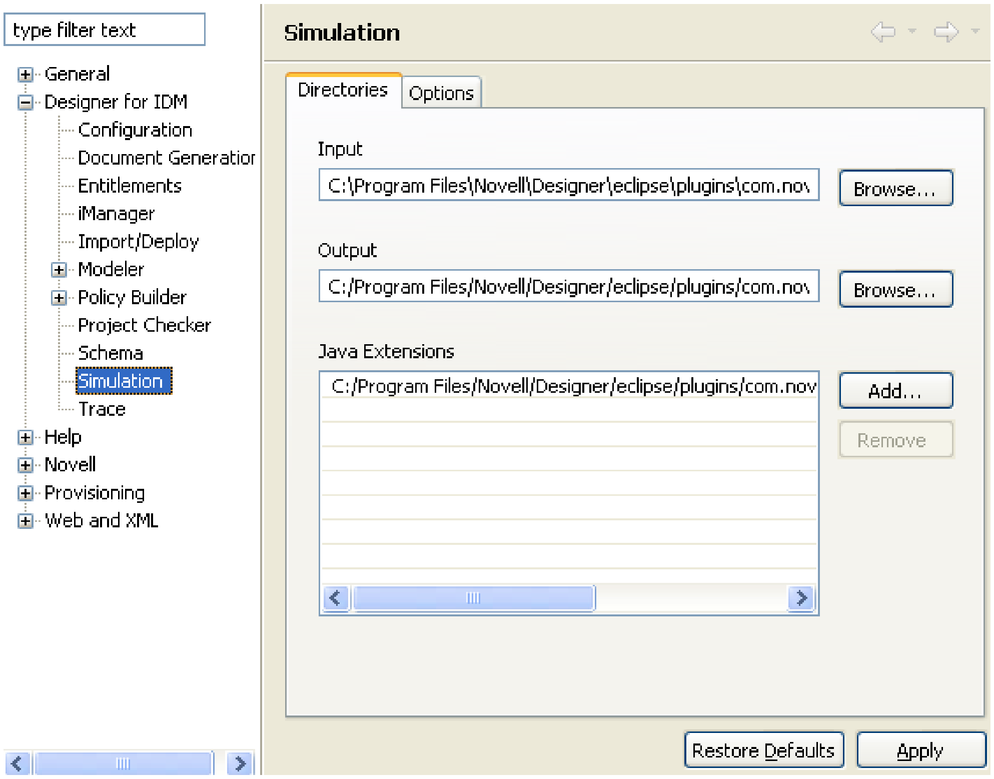
Policies that contain references to external Java® extensions can now be simulated by specifying the directory where the `.jar` file is located.

To determine or change the extension directory:

1 Select *Windows* > *Preferences* from the tool bar.

2 Navigate to the *Designer for IDM* > *Simulation* page.

3 Copy the jar file containing the Java class to the specified directory and simulate the policy.



Simulation

Input

C:\Program Files\Novell\Designer\eclipse\plugins\com.nov

Output

C:/Program Files/Novell/Designer/eclipse/plugins/com.nov

Java Extensions

C:/Program Files/Novell/Designer/eclipse/plugins/com.nov

4 Click *Apply* to save your changes, or click *OK* to save your changes and close the window.

NOTE: The *Enable unsupported and experimental pre-release functionality* option enables the Policy Simulator to test the policies against a live Identity Vault or the connected systems. This option is not supported in Designer 2.1 and is not documented.

Designer allows you to specify more than one directory that contains the external Java classes. To specify an additional directory:

1 Click *Add*.

2 Browse to and select the desired directory, then click *OK*.

3 To remove a directory, click *Remove*.

Storing Information in Resource Objects

10

Resource objects store information that drivers use. The resource objects can hold arbitrary data in any format. Novell® Identity Manager 3.5.1 contains different types of resource objects.

- Section 10.1, "Generic Resource Objects," on page 143
- Section 10.2, "Mapping Table Objects," on page 145
- Section 10.3, "ECMAScript Objects," on page 148
- Section 10.4, "Application Objects," on page 148
- Section 10.5, "Repository Objects," on page 149
- Section 10.6, "Library Objects," on page 149

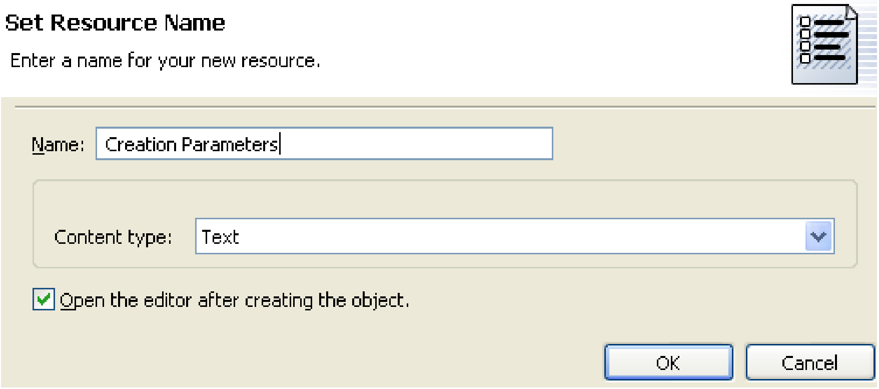
10.1 Generic Resource Objects

Generic Resource objects allow you 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.

- Section 10.1.1, "Creating a Resource Object," on page 143
- Section 10.1.2, "Using a Generic Resource Object," on page 144

10.1.1 Creating a Resource Object

1. In the *Outline* view, right-click on 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*.



Set Resource Name

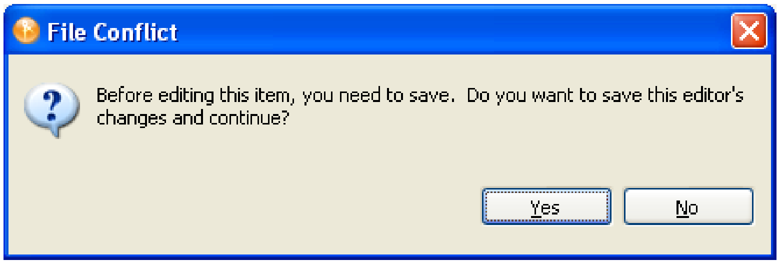
Enter a name for your new resource.

Name: Creation Parameters

Content type: Text

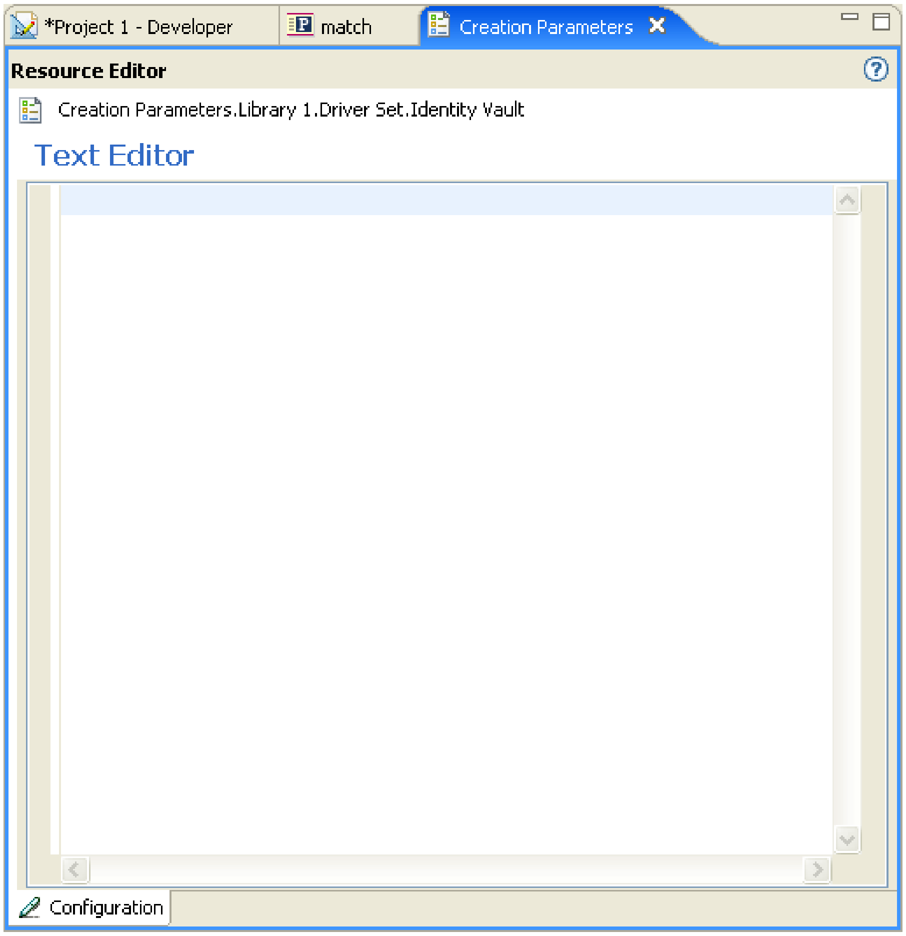
☑ Open the editor after creating the object.

5 Click *Yes* in the file conflict messages.



Before editing this item, you need to save. Do you want to save this editor's changes and continue?

6 Specify the desired text or XML, then press Ctrl+S to save the resource object.



Creation Parameters.Library 1.Driver Set.Identity Vault

10.1.2 Using a Generic Resource Object

A resource object is a box to store information. It is an eDirectory object and to use the information in the object, you treat it as any other 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 319) or Destination Attribute (page 297) tokens. To write information to the object, use the following actions:

- Clear Destination Attribute Value (page 227)
- Clear Source Attribute Value (page 229)
- Set Default Attribute Value (page 261)
- Set Source Attribute Value (page 274)

10.2 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 339) 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. Section 10.2.1, “Creating a Mapping Table Object,” on page 145
2. Section 10.2.2, “Adding a Mapping Table Object to a Policy,” on page 147

To edit a mapping table, see Section 10.2.3, “Editing a Mapping Table Object,” on page 147.

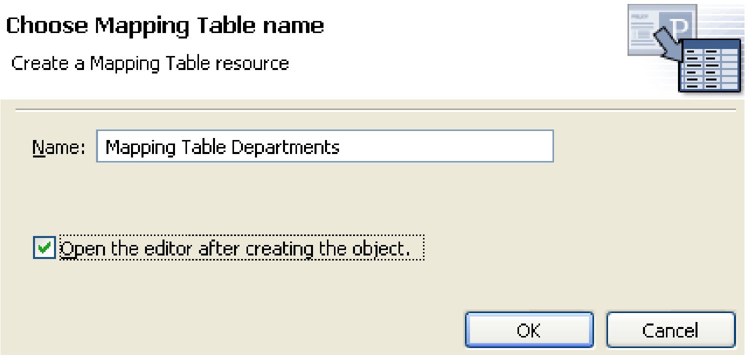
10.2.1 Creating a Mapping Table Object

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.

3 Select the check box for *Open the editor after creating the object*, then click *OK*.



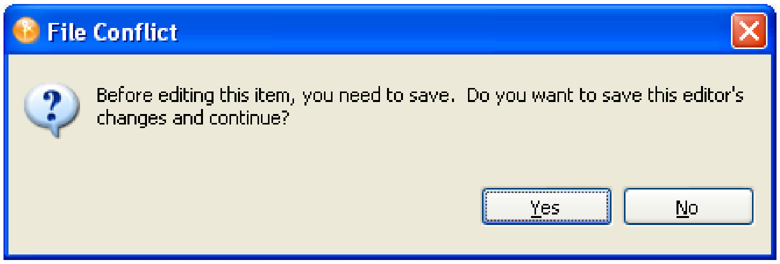
Choose Mapping Table name

Create a Mapping Table resource

Name: Mapping Table Departments

☑ Open the editor after creating the object.

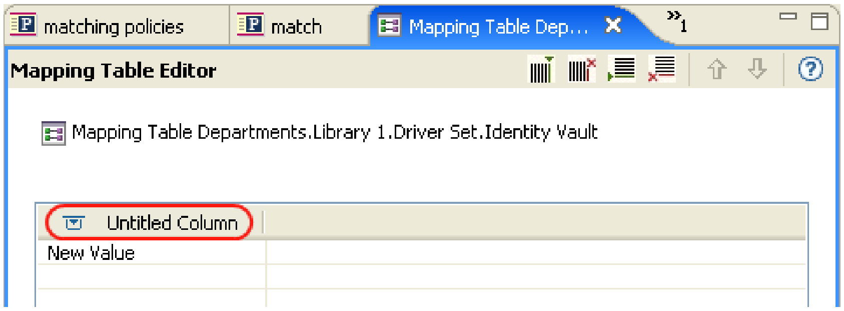
4 Click *Yes* in the file conflict message to save the mapping table.



File Conflict

Before editing this item, you need to save. Do you want to save this editor's changes and continue?

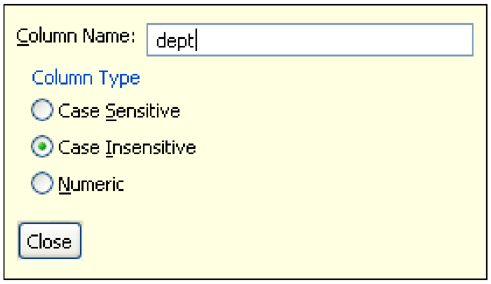
5 Click the untitled column.



Mapping Table Departments.Library 1.Driver Set.Identity Vault

Untitled Column	
New Value	

6 Specify the name of the column, then select if the value is *Case Sensitive*, *Case Insensitive*, or *Numeric*.



Column Name: dept

Column Type

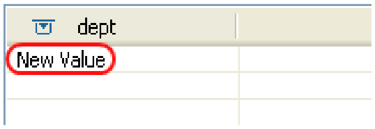
○ Case Sensitive

● Case Insensitive

○ Numeric

7 Click *Close*.

8 Click *New Value*, then specify the value for the row.



dept	
New Value	

9 (Optional) To add an addition column, right-click in the Mapping Table editor, then select *Add Column*.

or

Click the *Add Column* icon, then repeat Step 5 through Step 7 on page 146.

10 (Optional) To add an additional row, right-click in the Mapping Table editor, then select *Add Row*.

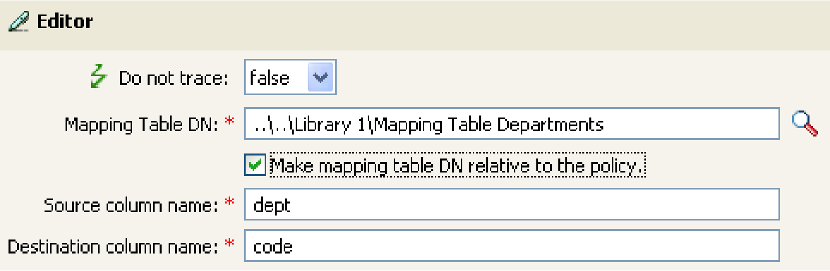
or

Click the *Add Row* icon, then repeat Step 8.

11. Press Ctrl+S to save the mapping table object.
12. Continue with Section 10.2.2, “Adding a Mapping Table Object to a Policy,” on page 147.

10.2.2 Adding a Mapping Table Object to a Policy

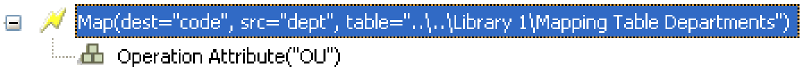
1. Either create a policy to use the mapping table in, or select an existing policy to edit.
2. Launch the Argument Builder in the Policy Builder.
3. Double-click *Map* from the list of Verbs to add it to the expression panel.
4. Select either *true* or *false* to indicate whether you want this mapping table traced.
5. In the Editor field, browse to and select the mapping table object created in Section 10.2.1, “Creating a Mapping Table Object,” on page 145.
6. Specify the source column name.
7. Specify the destination column name.



Editor

Do not trace:	false
Mapping Table DN: *	..\..\Library 1\Mapping Table Departments
	☑ Make mapping table DN relative to the policy.
Source column name: *	dept
Destination column name: *	code

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.

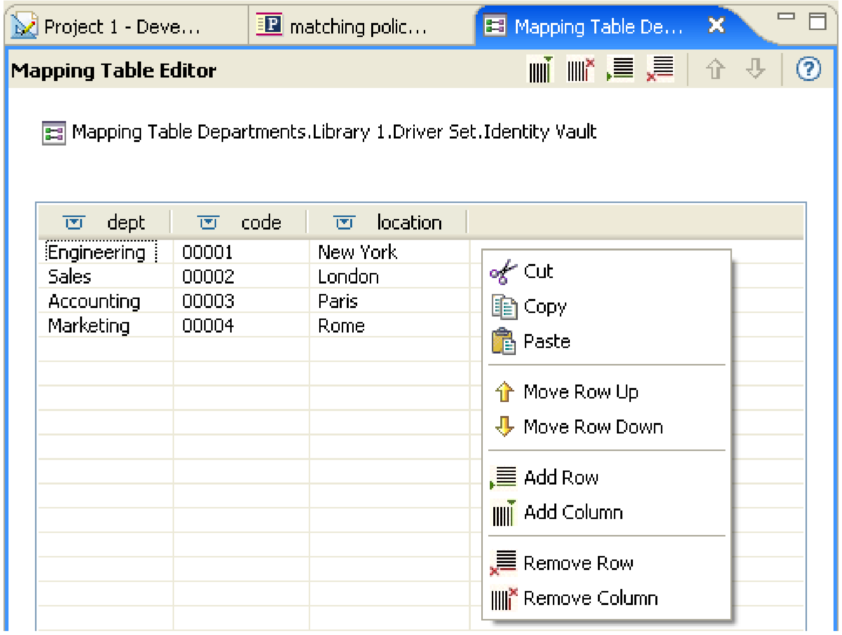


```
Map(dest="code", src="dept", table="..\..\Library 1\Mapping Table Departments")
    Operation Attribute("OU")
```

10.2.3 Editing a Mapping Table Object

Designer provides the following options to edit the mapping table:

Figure 10-1 *Editing a Mapping Table*



Mapping Table Departments.Library 1.Driver Set.Identity Vault

dept	code	location
Engineering	00001	New York
Sales	00002	London
Accounting	00003	Paris
Marketing	00004	Rome

- **Cut:** Cuts the selected row.
- **Copy:** Copies the selected row.
- **Paste:** Pastes the selected row.
- **Move Row Up:** Moves the selected row up one row.
- **Move Row Down:** Moves the selected row down one row.
- **Add Row:** Adds a row to the mapping table.
- **Add Column:** Adds a column to the mapping table.
- **Remove Row:** Deletes a row from the mapping table.
- **Remove Column:** Deletes a column from the mapping table.

10.2.4 Testing a Mapping Table Object

You can test the functionality of the mapping table with the Policy Simulator. The Policy Simulator tests the mapping table by testing the policy that is using the mapping table. For more information, see Chapter 9, “Testing Policies with the Policy Simulator,” on page 135.

10.3 ECMAScript Objects

ECMAScript objects are resource objects that store ECMAScripts. The ECMAScript is used by policies and style sheets. For more information on ECMAScript, see Chapter 11, “Using ECMAScript in Policies,” on page 153.

10.4 Application Objects

Application objects store authentication parameter values for Novell Credential Provisioning policies. There application objects for Novell SecureLogin and Novell SecretStore®. For information on how to create application objects for SecureLogin, see "Creating an Application Object for Novell SecureLogin" in *Novell Credential Provisioning Policies for Identity Manager 3.5.1*. For information on how to create application objects for SecretStore, see "Creating an Application Object for Novell SecretStore" in *Novell Credential Provisioning Policies for Identity Manager 3.5.1*.

10.5 Repository Objects

Repository objects store static configuration information for Novell Credential Provisioning policies. There are repository objects for Novell SecureLogin and Novell SecretStore. For information on how to create repository objects for SecureLogin, see "Creating a Repository Object for Novell SecureLogin " in *Novell Credential Provisioning Policies for Identity Manager 3.5.1*. For information on how to create repository objects for SecretStore, see "Creating a Repository Object for Novell SecretStore" in *Novell Credential Provisioning Policies for Identity Manager 3.5.1*.

10.6 Library Objects

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 eDirectory™ container. Multiple libraries can exist in an 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.

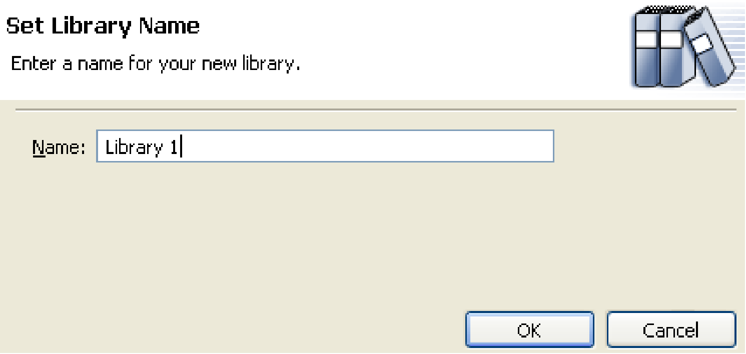
Style sheets, policies, rules, and other resource objects can be stored in a library and be referenced by one or more drivers.

- Section 10.6.1, "Creating Library Objects," on page 149
- Section 10.6.2, "Adding Policies to the Library Objects," on page 150
- Section 10.6.3, "Using Policies in the Library Objects," on page 150

10.6.1 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*.



Set Library Name

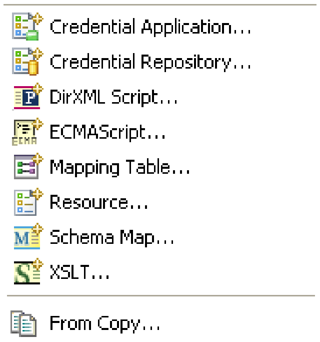
Enter a name for your new library.

Name: Library 1

10.6.2 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, then select *New* and what ever type of object you want stored in the library. The options are:



- **Credential Application:** Stores application authentication parameter values for Novell Credential Provisioning policies. For information, see *Novell Credential Provisioning Policies for Identity Manager 3.5.1*.
- **Credential Repository:** Stores static configuration information for Novell Credential Provisioning policies. For information, see *Novell Credential Provisioning Policies for Identity Manager 3.5.1*.
- **DirXML Script:** Creates a policy set. See Section 3.3, "Creating a Policy," on page 26 for more information.
- **ECMAScript:** Creates an ECMAScript object. See Section 11.1, "Creating an ECMAScript Object," on page 153 for more information.
- **Mapping Table:** Creates a mapping table object. For more information, see Section 10.2, "Mapping Table Objects," on page 145.
- **Resource:** Creates a generic resource object. For more information, see Section 10.1, "Generic Resource Objects," on page 143.
- **Schema Map:** Creates a Schema Map object. For more information, see Chapter 6, "Defining Schema Mapping Policies," on page 73.
- **XSLT:** Creates an XSLT style sheet in the library. For more information, see "Defining Policies by Using XSLT Style Sheets" in *Understanding Policies for Identity Manager 3.5.1*.
- **From Copy:** Creates a copy of an existing object.

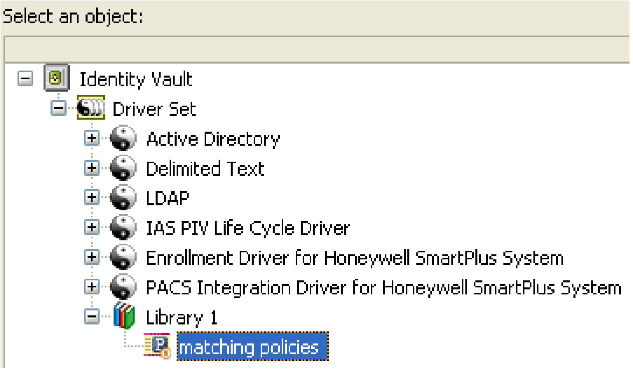
10.6.3 Using Policies in the Library Objects

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.



Select an object:

- Identity Vault
  - Driver Set
    - Active Directory
    - Delimited Text
    - LDAP
    - IAS PIV Life Cycle Driver
    - Enrollment Driver for Honeywell SmartPlus System
    - PACS Integration Driver for Honeywell SmartPlus System
    - Library 1
      - matching policies

Using ECMAScript in Policies 11

ECMAScript is a scripting programming language, standardized by Ecma International. It is often referred to as JavaScript® or JScript*, but these are implementations of ECMAScript. Identity Manager 3.5.1 supports a new object type called ECMAScript objects. 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 Custom Forms (http://www.novell.com/documentation/idm35/dgpro/data/prdefcreateformschapter.html).

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 the ECMAScript Language Specification (http://www.ecma-international.org/publications/standards/Ecma-262.htm) for information on how to use the ECMAScript language.

- Section 11.1, "Creating an ECMAScript Object," on page 153
- Section 11.2, "Using the ECMAScript Editor," on page 154
- Section 11.3, "Examples of ECMAScripts with Policies," on page 162
- Section 11.4, "Changing JavaScript Files Preferences," on page 165

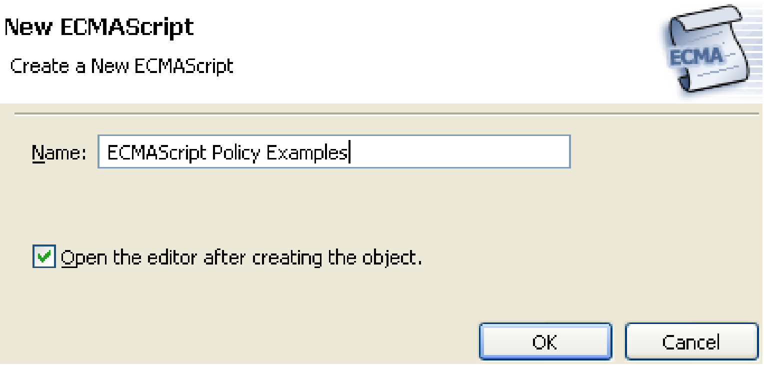
11.1 Creating an ECMAScript Object

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*.

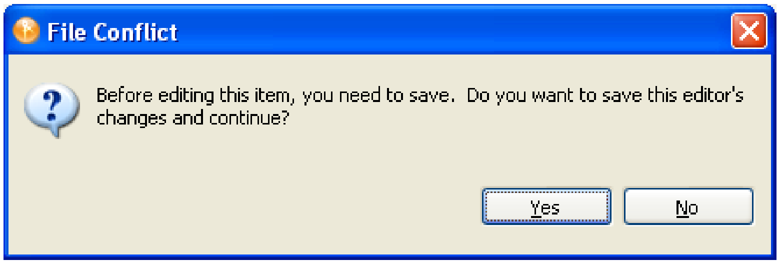

New ECMAScript

Create a New ECMAScript

Name: ECMAScript Policy Examples

☑ Open the editor after creating the object.

4 Click *Yes* in the file conflict message to save the ECMAScript object.



Before editing this item, you need to save. Do you want to save this editor's changes and continue?

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 11.2, “Using the ECMAScript Editor,” on page 154.

11.2 Using the ECMAScript Editor

ECMAScript objects are supported only with servers that have Identity Manager 3.5.1 version. If a server in a selected driver set is earlier than Identity Manager 3.5.1, an error message is displayed, and Designer does not allow the object to be created. Change the version of the server to Identity Manager 3.5.1 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*.

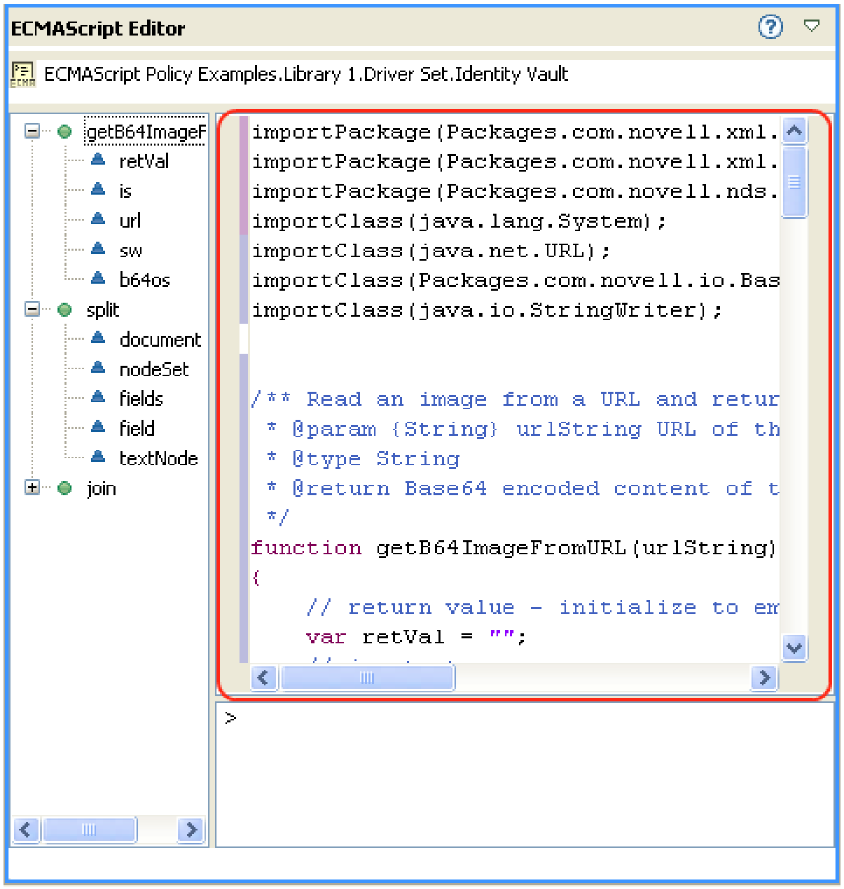
The ECMAScript editor provides different types of functionality depending upon which section you are using.

- Section 11.2.1, “Main Scripting Area,” on page 154
- Section 11.2.2, “Expression Builder,” on page 157
- Section 11.2.3, “Functions and Variables,” on page 159
- Section 11.2.4, “Error Display,” on page 160
- Section 11.2.5, “Shell Area,” on page 161

11.2.1 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 11-1 *Main Scripting Area*



```
importPackage(Packages.com.novell.xml.
importPackage(Packages.com.novell.xml.
importPackage(Packages.com.novell.nds.
importClass(java.lang.System);
importClass(java.net.URL);
importClass(Packages.com.novell.io.Bas
importClass(java.io.StringWriter);


/** Read an image from a URL and retur
 * @param {String} urlString URL of th
 * @type String
 * @return Base64 encoded content of t
 */
function getB64ImageFromURL(urlString)
{
    // return value - initialize to em
    var retVal = "";
```

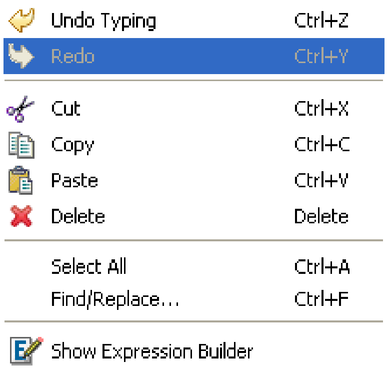
- "Using an Existing ECMAScript" on page 155
- "Editing an ECMAScript" on page 156
- "Coding Help for ECMAScript" on page 156

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

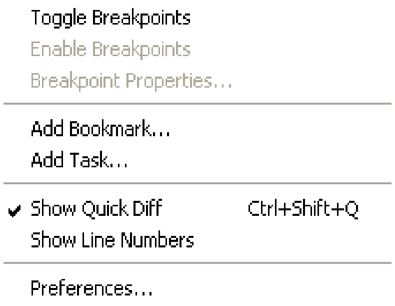
1 Right-click in the main scripting area, then select the desired option.



- **Undo Typing:** Undoes the typing that has occurred.
- **Redo:** Redoes the last action.
- **Cut:** Cuts the selected area.
- **Copy:** Copies the selected area.
- **Paste:** Pastes the information in the Clipboard into the main scripting area.
- **Delete:** Deletes the selected information from the main scripting area.
- **Select All:** Selects 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 Section 11.2.2, "Expression Builder," on page 157.

Coding Help for ECMAScript

1 Right-click in the left margin of the main scripting area, then select the desired option.



- **Toggle Breakpoints:** To be implemented.
- **Enable Breakpoints:** Set breakpoints in the ECMAScript.
- **Breakpoint Properties:** View the properties of the breakpoints.
- **Add Bookmark:** Places a bookmark icon on a line in the ECMAScript editor.
- **Add Task:** 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.

- **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`.

11.2.2 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.

   or

   Right-click the shell area of the ECMAScript editor.

To access the Expression Builder through the Policy Builder:

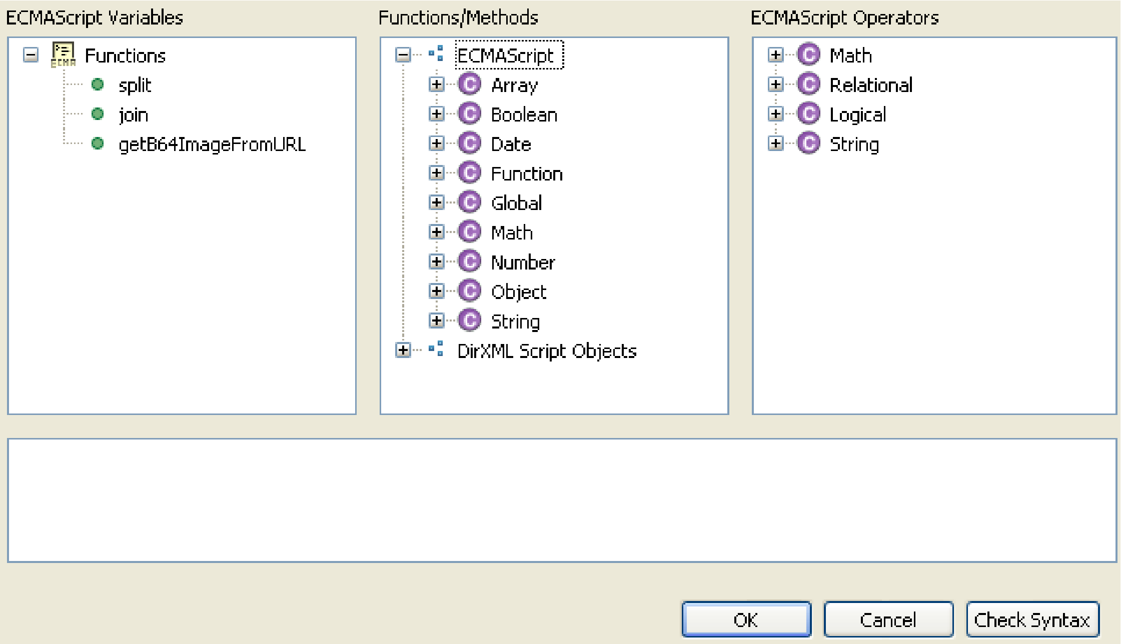
1. Click the *Launch ECMA Expression Builder* icon next to the following actions or conditions:.
   - XPath expression
   - append XML element
   - append XML text
   - clone by XPath expressions
   - set XML attribute
   - strip XPath expression

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 11-2 Expression Builder



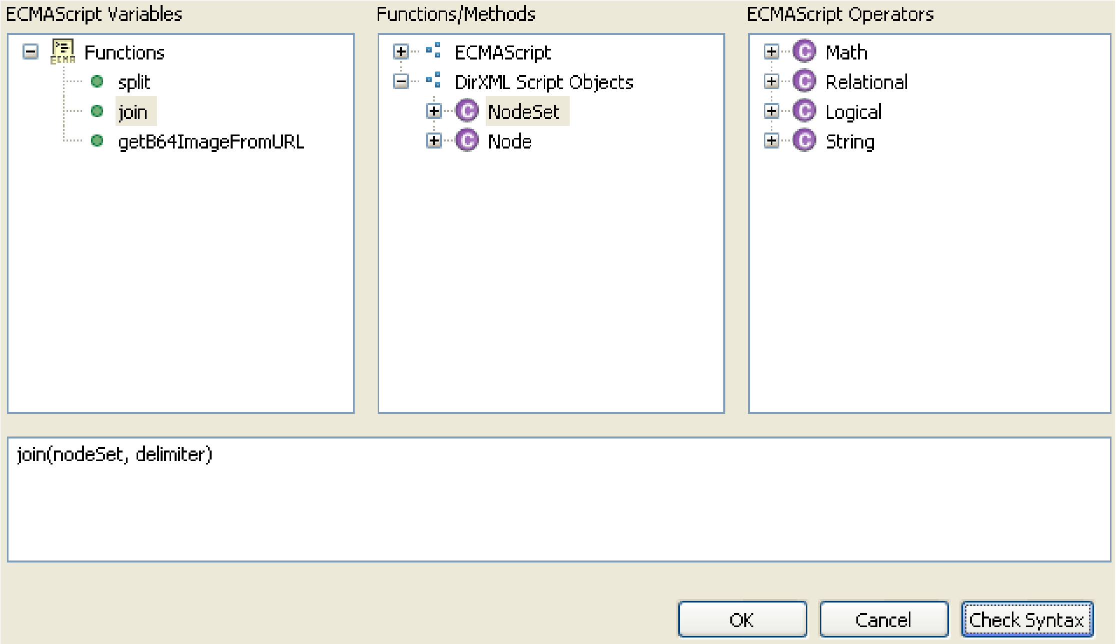
ECMAScript/Variables lists all of the current defined functions in the ECMAScript. *Function/ Methods* contains the standard ECMAScript functions and the DirXML Script functions. *ECMAScript Operators* displays the standard ECMAScript operators.

To use the Expression Builder:

1. (Optional) Click the desired *ECMAScript/Variables*.
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 NodeSet function or method, but there is no ECMAScript operator selected.

Figure 11-3 *Expression Builder Example*

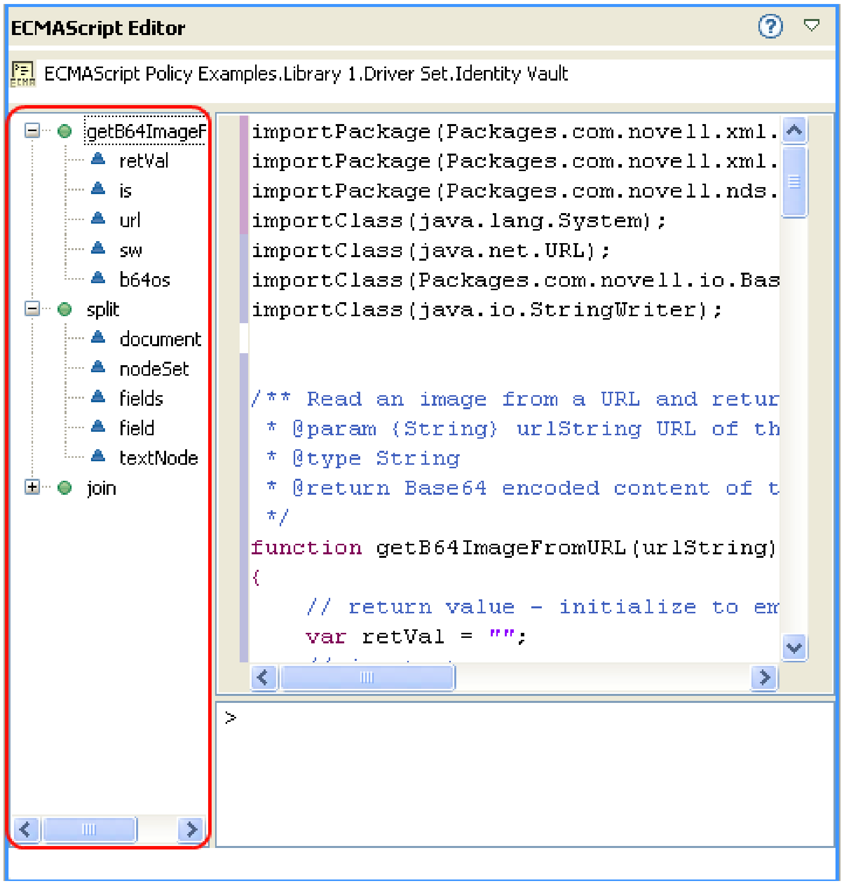


```
join(nodeSet, delimiter)
```

11.2.3 Functions and Variables

As functions and variables are defined in the ECMAScript, they are displayed on the left side of the ECMAScript editor.

Figure 11-4 *Functions and Variables*



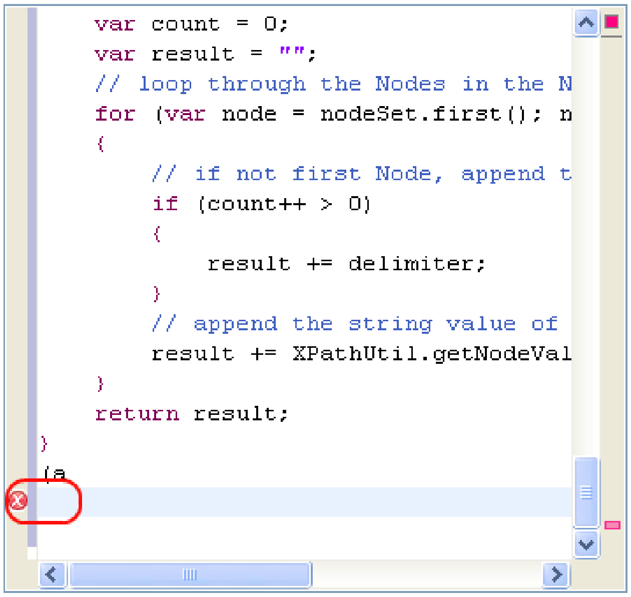
```
importPackage(Packages.com.novell.xml.
importPackage(Packages.com.novell.xml.
importPackage(Packages.com.novell.nds.
importClass(java.lang.System);
importClass(java.net.URL);
importClass(Packages.com.novell.io.Bas
importClass(java.io.StringWriter);


/** Read an image from a URL and retur
 * @param {String} urlString URL of th
 * @type String
 * @return Base64 encoded content of t
 */
function getB64ImageFromURL(urlString)
{
    // return value - initialize to em
    var retVal = "";
```

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).

11.2.4 Error Display

As the ECMAScript is created, errors are displayed in the main scripting area and in the *Problem* view. The main scripting area displays the errors as a red X on the line where the error occurs.

Figure 11-5 *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
```

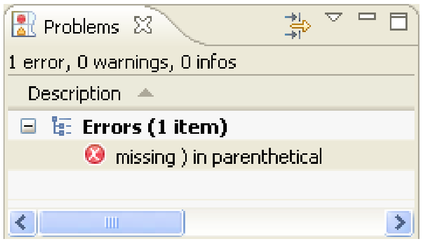
The *Problem* view accumulates the errors as the ECMAScript is typed, displays the cause of the error.

Double-click the error in the *Problem* view. The cursor jumps to the problem line in the main scripting area.

To access the *Problem* view:

1 In the toolbar select *Window* > *Show View* > *Other* > *General Problems*.

The *Problem* view is displayed below the ECMAScript editor.



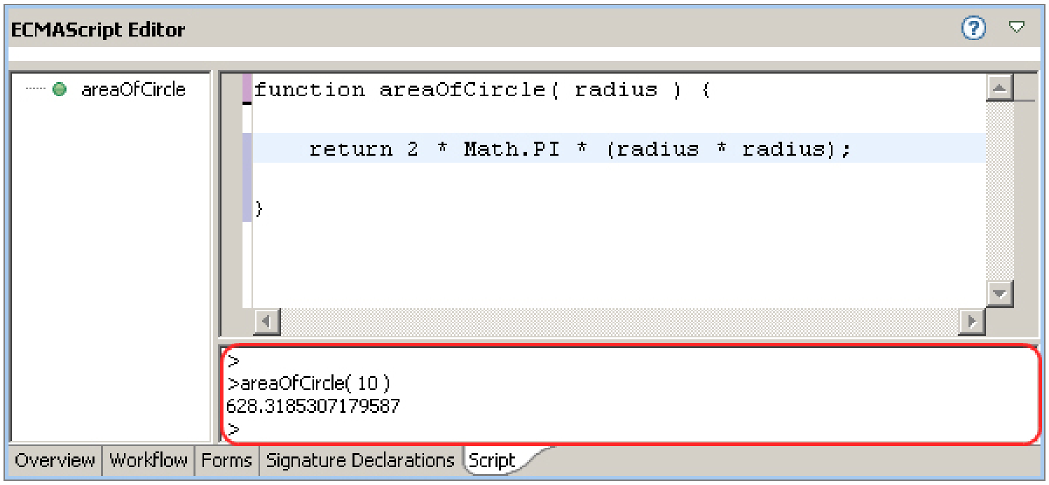
1 error, 0 warnings, 0 infos

Description
Errors (1 item)
missing) in parenthetical

11.2.5 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 11-6 *Shell Area*



```
function areaOfCircle( radius ) {

    return 2 * Math.PI * (radius * radius);

}
```

```
>
>areaOfCircle( 10 )
628.3185307179587
>
```

Figure 11-6 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.

11.3 Examples of ECMAScripts with Policies

The following examples use the ECMAScript file demo.js (../samples/demo.js) with different policies. The `demo.js` file contains three ECMAScript function definitions.

- Section 11.3.1, "DirXML Script Policy Calling an ECMAScript Function," on page 162
- Section 11.3.2, "XSLT Policy Calling an ECMAScript Function at the Driver Level," on page 163
- Section 11.3.3, "XSLT Policy Calling an ECMAScript Function in the Style Sheet," on page 165

11.3.1 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.policybu
ilder_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>
```

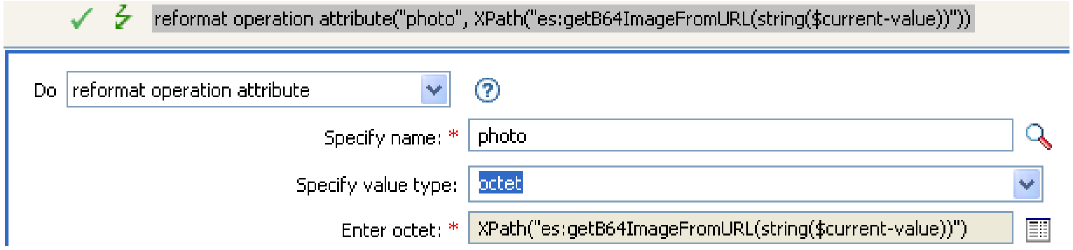
Function: `<static> String getB64ImageFromURL(<String> urlString)`

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 11-7** Reformat Photo Example*



reformat operation attribute("photo", XPath("es:getB64ImageFromURL(string($current-value))"))

Do: reformat operation attribute

Specify name: * photo

Specify value type: octet

Enter octet: * XPath("es:getB64ImageFromURL(string($current-value))")

The ECMAScript calls the getB64ImageFromURL function, which then returns the current value as a string.

11.3.2 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 164
- "Split" on page 164

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.

11.3.3 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.

11.4 Changing JavaScript Files Preferences

Designer allows you to change how JavaScript files are displayed and used. ECMAScript is a JavaScript and these preferences affect the ECMAScript editor. To change the preferences:

1. In the Designer toolbar, select *Window* > *Preferences* > *Web and XML* > *JavaScript Files*.
2. Change the desired settings, then click *OK*.

   See Section 11.4.1, “JavaScript Files Preferences,” on page 165 for a list of all of the preferences.

11.4.1 JavaScript Files Preferences

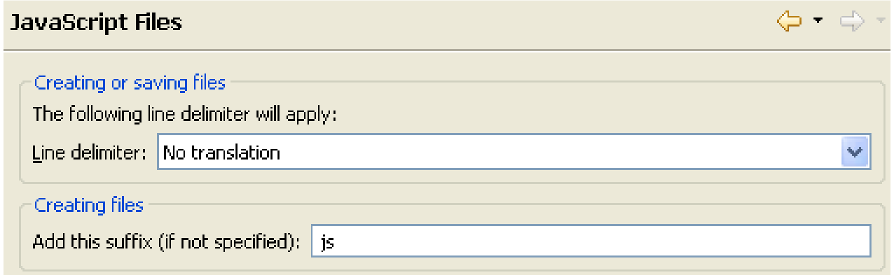
Changes how JavaScript files are handled by Designer. There are multiple options to change.

- “JavaScript Files” on page 165
- “JavaScript Files > JavaScript Source” on page 166
- “JavaScript Files > JavaScript Styles” on page 167
- “JavaScript Validation” on page 168

JavaScript Files

Changes how JavaScript files are created.

Figure 11-8 *JavaScript Files*



JavaScript Files

Creating or saving files

The following line delimiter will apply:

Line delimiter: No translation

Creating files

Add this suffix (if not specified): js

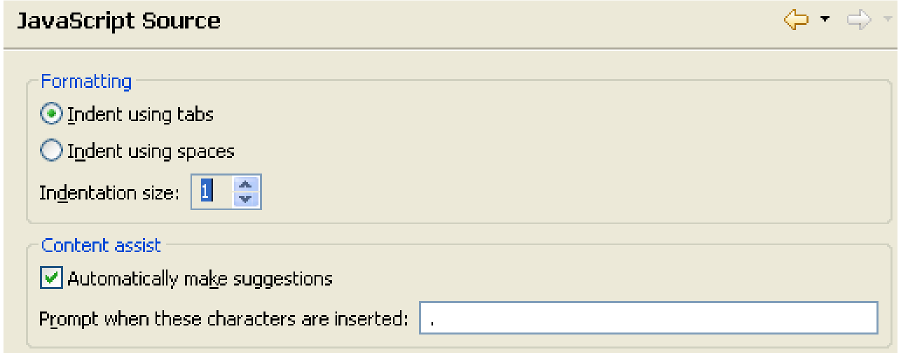
Table 11-1 *JavaScript Files*

Setting	Description
Creating or saving files: Line delimiter	Sets what type of line delimiter is applied to the file. The options are: ◆ *No translation* ◆ *UNIX* ◆ *Mac* ◆ *Windows*
Creating files: Add this suffix (if not specified)	Sets the suffix to file. The default value is js. It can be set to any value.

JavaScript Files > JavaScript Source

Changes the formatting for the JavaScript files.

Figure 11-9 *JavaScript Files > JavaScript Source*



JavaScript Source

Formatting

● Indent using tabs

○ Indent using spaces

Indentation size: 1

Content assist

☑ Automatically make suggestions

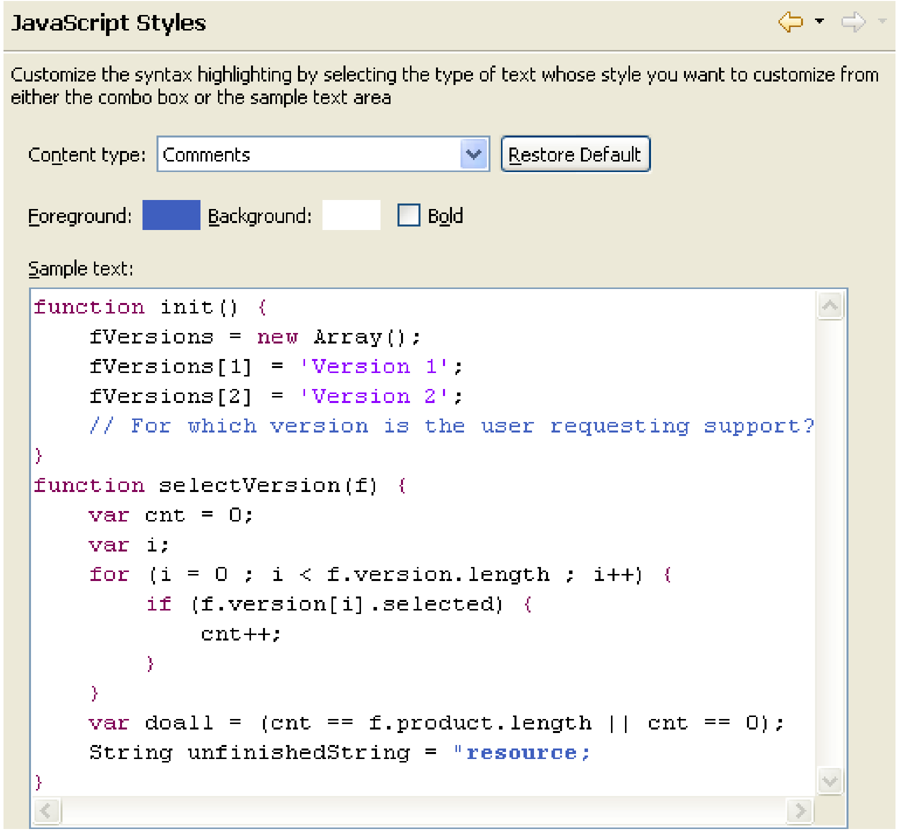
Prompt when these characters are inserted: .

Table 11-2 JavaScript Files > JavaScript Source

Setting	Description
Formatting	Sets the formatting for the editor. The options are: ◆ *Indent using tabs* ◆ *Indent using spaces* ◆ *Indention size*: Changes the indention size by setting a numeric value.
Content assist	Helps with prompts when creating the files. *Automatically make suggestions*: Can be enabled or disabled if the check box is selected. *Prompt when these characters are inserted*: Allows you to receive prompts when the specified characters are entered.

JavaScript Files > JavaScript Styles

Customizes the content of the JavaScript files.



JavaScript Styles

Customize the syntax highlighting by selecting the type of text whose style you want to customize from either the combo box or the sample text area

Content type: Comments ▾ [Restore Default]

Foreground: 🟦 Background: ⬜ ☐ Bold

Sample text:

```
function init() {
    fVersions = new Array();
    fVersions[1] = 'Version 1';
    fVersions[2] = 'Version 2';
    // For which version is the user requesting support?
}
function selectVersion(f) {
    var cnt = 0;
    var i;
    for (i = 0 ; i < f.version.length ; i++) {
        if (f.version[i].selected) {
            cnt++;
        }
    }
    var doall = (cnt == f.product.length || cnt == 0);
    String unfinishedString = "resource;
}
```

Figure 11-10 JavaScript Files > JavaScript Styles

Table 11-3 *JavaScript Files > JavaScript Styles*

Setting	Description
Content type	Selects the content to be customized. You can change each element independently. The sections that can be changed are: ◆ *Comments* ◆ *Default Code* ◆ *Keywords* ◆ *Literal Strings* ◆ *Unfinished Strings and Comments* If you select the element in the sample text field, the content type changes to what is selected.
Foreground	Displays the color that is set for the foreground. Double-click the color field to select another color.
Background	Displays the color that is set for the background. Double-click the color field to select another color.
Bold	Allows you to bold an element. Select the element, the click the *Bold* check box.
Sample text	Displays a sample file to see the changes.

JavaScript Validation

Allows the editor to validate the JavaScript as it is entered. Select *Automatically validate scripts* to automatically validate the scripts. If it is not select the JavaScript will not be validated.

Conditions

12

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.

This section contains detailed information about all conditions that are available through the Policy Builder interface.

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

Select the condition test type.

Operator	Returns True when...
Associated	There is an established association for the current object.
Not Association	There is not an established association for the current object.
Available	There is a non-empty association value specified by the current operation.
Not available	The association is not available for the current object.
Equal	The association value specified by the current operation is exactly equal to the content of the if association.
Not Equal	The association value specified by the current operation is not 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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
Less Than	The association value specified by the current operation is less than the content of the condition when compared using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

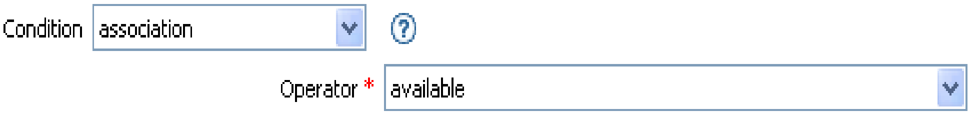
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

This example tests to see if the association is available. When this condition is met, the actions that are defined are executed.



Condition association

Operator * available

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.

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.
Not Available	Available would return False.
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 using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
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 using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

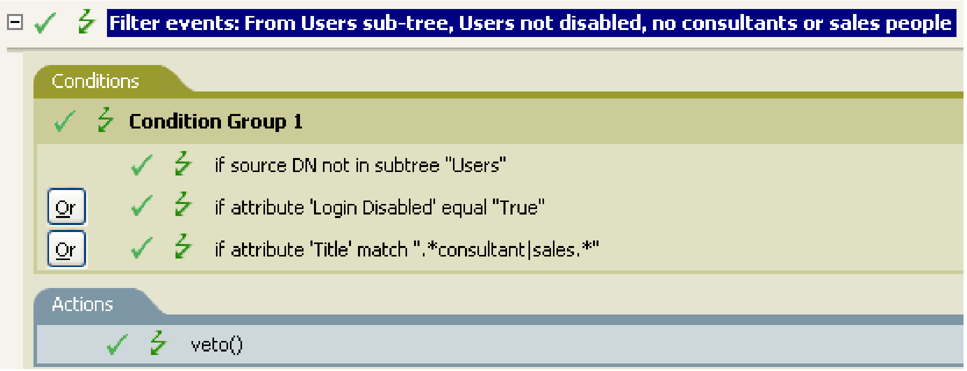
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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 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 Novell® Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Event-FilterByContainerDisabledOrTitle.xml (../samples/001-Event-FilterByContainerDisabledOrTitle.xml).



Filter events: From Users sub-tree, Users not disabled, no consultants or sales people

Conditions

Condition Group 1

if source DN not in subtree "Users"

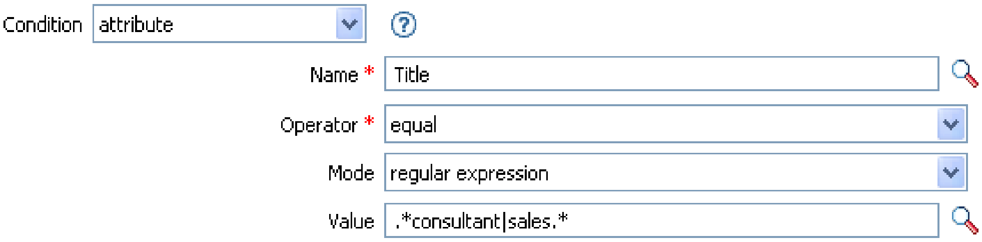
Or if attribute 'Login Disabled' equal "True"

Or if attribute 'Title' match ".*consultant|sales.*"

Actions

veto()

The condition is looking for any User object that has an attribute of Title with a value of consultant or sales.



Condition: attribute

Name *: Title

Operator *: equal

Mode: regular expression

Value: .*consultant|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.
Not Available	Available would return False.
Equal	There is an object class name available in the current operation, and it equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
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 using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less 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.

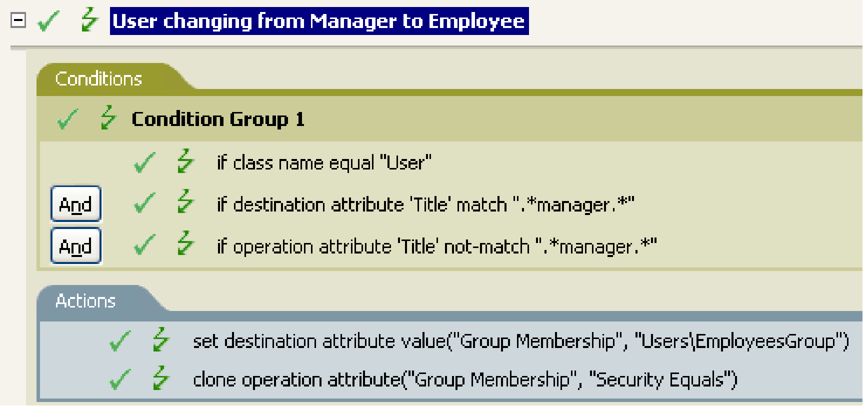
Mode	Description
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun’s Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 004-Command-GroupChangeOnTitleChange.xml (../samples/004-Command-GroupChangeOnTitleChange.xml).



User changing from Manager to Employee

Conditions

Condition Group 1

if class name equal "User"

And if destination attribute 'Title' match ".*manager.*"

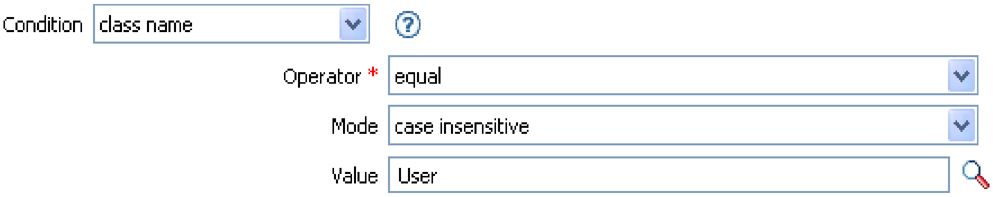
And if operation attribute 'Title' not-match ".*manager.*"

Actions

set destination attribute value("Group Membership", "Users\EmployeesGroup")

clone operation attribute("Group Membership", "Security Equals")

Checks to see if the class name of the current object is User.



Condition: class name

Operator *: equal

Mode: case insensitive

Value: User

If Destination Attribute

Performs a test on attribute values of the current object in the destination data store. The test performed depends on the specified operator.

Fields

Name

Specify the name of the attribute to test.

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.
Not Available	Available would return False.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode. If `mode="structured"`, the content must be a set of `<component>` elements; otherwise, it must be text.
Not Great Than	Greater Than or Equal would return False.
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 using the specified comparison mode. If `mode="structured"`, the content must be a set of `<component>` elements; otherwise, it must be text.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

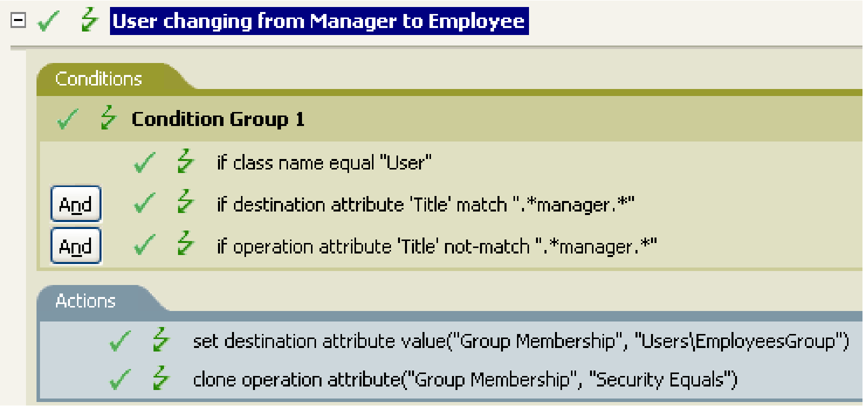
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

The example uses the condition If Attribute 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 004-CommandGroupChangeOnTitleChange.xml (../samples/004-Command-GroupChangeOnTitleChange.xml).



User changing from Manager to Employee

Conditions

Condition Group 1

if class name equal "User"

And if destination attribute 'Title' match ".*manager.*"

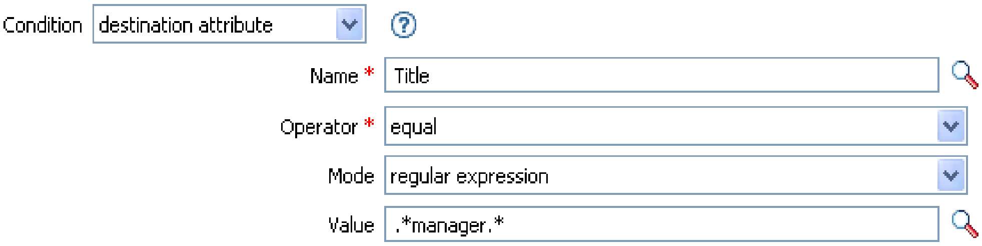
And if operation attribute 'Title' not-match ".*manager.*"

Actions

set destination attribute value("Group Membership", "Users\EmployeesGroup")

clone operation attribute("Group Membership", "Security Equals")

The policy checks to see if the value of the title attribute contains manager.



Condition	destination attribute
Name *	Title
Operator *	equal
Mode	regular expression
Value	.*manager.*

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.

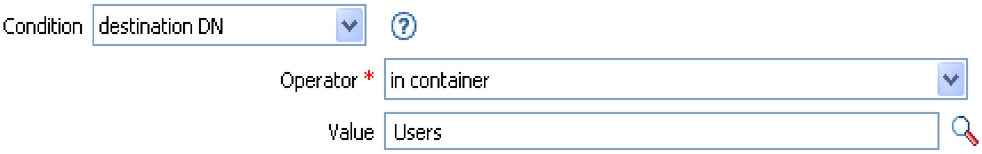
Operator	Returns True when...
Available	There is a destination DN available.
Not Available	Available would return False.
Equal	There is a destination DN available, and it equals the specified value when compared using semantics appropriate to the DN format of the destination data store.
Not Equal	Equal would return False.
In Container	There is a destination DN available, and it represents an object in the container, specified by value, when compared using semantics appropriate to the DN format of the destination data store.
Not in Container	In Container would return False.
In Subtree	There is a destination DN available, and it represents an object in the subtree, specified by value, when compared using semantics appropriate to the DN format of the destination data store.
Not in Subtree	In Subtree would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- In Container
- Not in Container
- In Subtree
- 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.

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.
Not available	Available would return False.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
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 using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.
Changing	The current operation contains a change (modify attribute or add attribute) of the named entitlement.
Not Changing	Changing would return False.
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 using the specified comparison mode.
Not Changing From	Changing From would return False.
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 using the specified comparison mode.
Not Changing To	Changing To would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Changing To
- Changing From
- Not Changing To
- Not Changing From
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

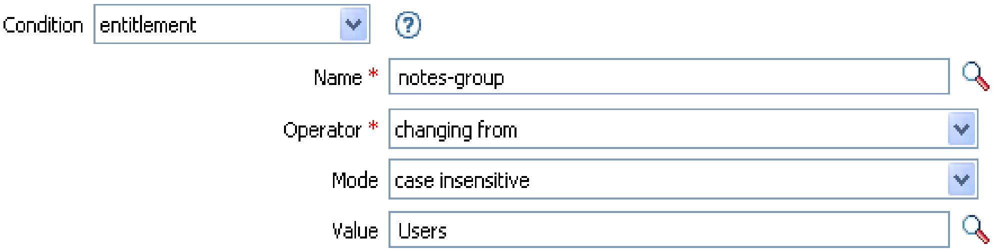
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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
- Changing To
- Changing From
- Not Changing To
- Not Changing From

- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example



Condition entitlement

Name * notes-group

Operator * changing from

Mode case insensitive

Value Users

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 Policies Components (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policycomponents.html).

Fields

Name

Specify the name of the global value to test for the selected condition.

Operator

Select the condition test type.

Operator	Returns True when...
Available	There is a global configuration value with the specified name.
Not Available	Available would return False.
Equal	There is a global configuration value with the specified name, and its value equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
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 using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than

- Not Less Than

Comparison Mode

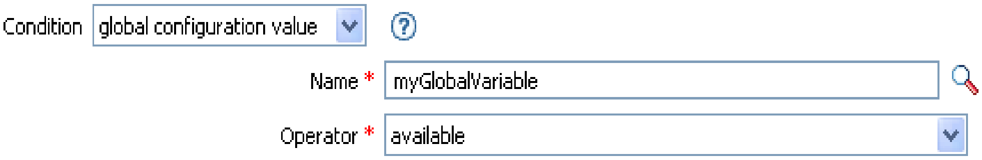
The condition has a comparison mode parameter that indicates how a comparison is done.

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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



Condition: global configuration value

Name *: myGlobalVariable

Operator *: available

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 Policies Components (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policycomponents.html).

Fields

Name

Specify the name of the local variable to test for the selected condition.

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.
Not Available	Available would return False.
Equal	There is a local variable with the specified name, and its value equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
Greater Than	There is a local variable with the specified name, and its value is greater than the content of the condition when compared using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
Less Than	There is a local variable with the specified name, and its value is less than the content of the condition when compared using the specified comparison mode.
Not Less Than	Less than or equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than

- Not Less Than

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

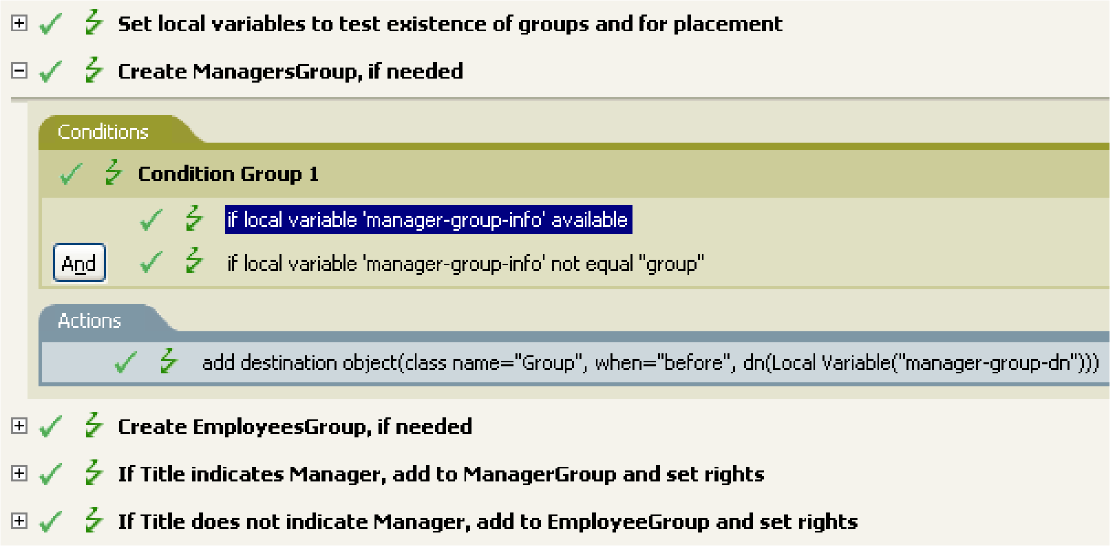
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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 group, Employee or Manager, 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html).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

Conditions

✓ Condition Group 1

✓ if local variable 'manager-group-info' available

And ✓ if local variable 'manager-group-info' not equal "group"

Actions

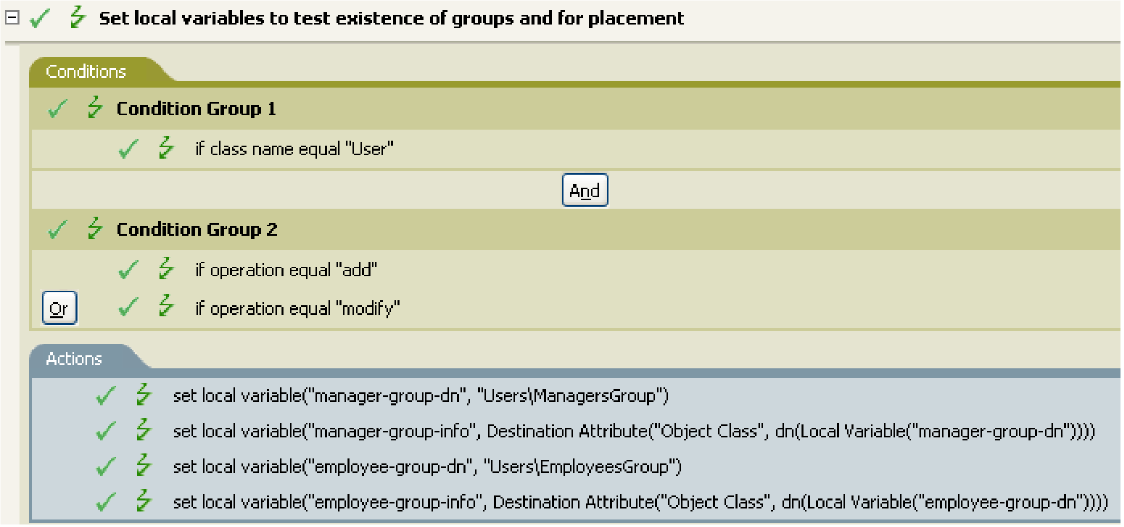
✓ add destination object(class name="Group", when="before", dn(Local Variable("manager-group-dn")))

⊞ ✓ 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

Conditions

✓ Condition Group 1

✓ if class name equal "User"

And

✓ Condition Group 2

✓ if operation equal "add"

Or ✓ if operation equal "modify"

Actions

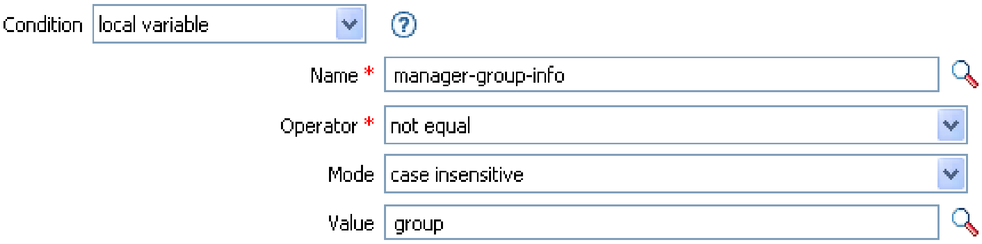
✓ set local variable("manager-group-dn", "Users\ManagersGroup")

✓ 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 is looking 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

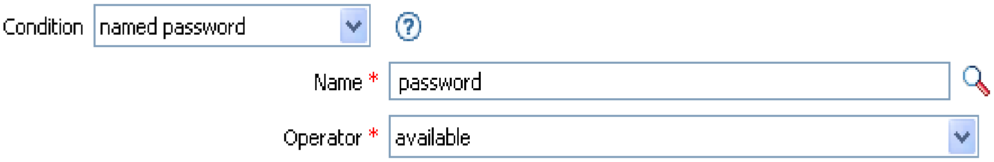
Specify the name of the named password to test for the selected condition.

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



Condition: named password

Name *: password

Operator *: available

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.

Operator

Select the condition test type.

Operator Returns True when...

Operator	Returns True when...
Available	There is a value available in the current operation other than a remove value for the specified attribute.
Not Available	Available would return False.
Equal	There is a value available in the current operation other than a remove value for the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode. If `mode="structured"`, the content must be a set of `<component>` elements; otherwise, it must be text.
Not Greater Than	Greater Than or Equal would return False.
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 using the specified comparison mode. If `mode="structured"` then the content must be a set of `<component>` elements; otherwise, it must be text.
Not Less Than	Less Than or Equal would return False.
Changing	The current operation contains a change for the specified attribute.
Not Changing	Changing would return False.
Changing From	The current operation contains a change that removes a value other than a remove value of the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Changing From	Changing From would return False.
Changing To	The current operation contains a change that adds a value other than a remove value to the specified attribute. It equals the specified value when compared using the specified comparison mode.

Operator	Returns True when...
Not Changing To	Changing To would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Changing To
- Changing From
- Not Changing To
- Not Changing From
- Greater Than
- Not Greater Than
- Less 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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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.

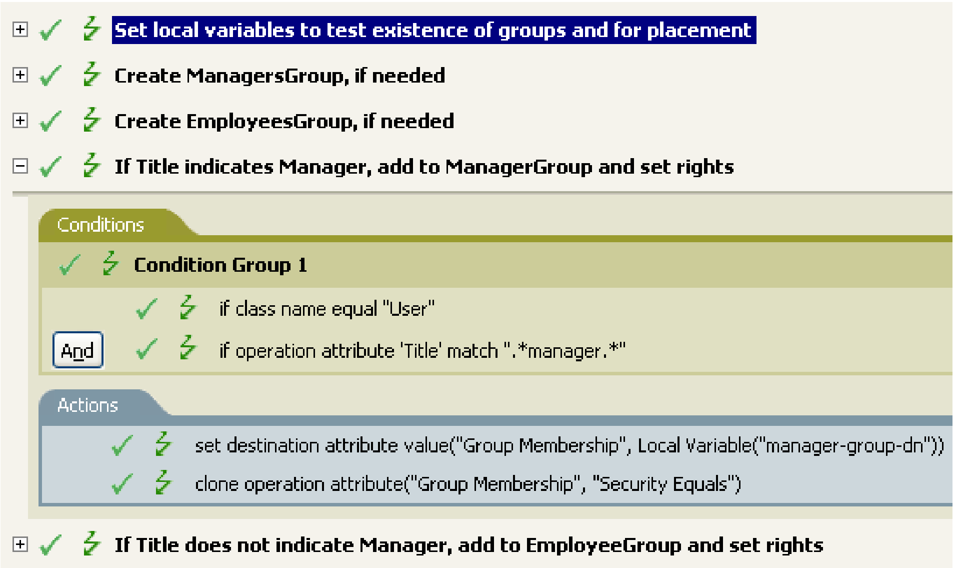
The operators that contain the comparison mode parameter are:

- Equal

- Not Equal
- Changing To
- Changing From
- Not Changing To
- Not Changing From
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

The example adds a User object to the appropriate group, Employee or Manager, 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 003-Command-Add-CreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).



⊞ ✓ Set local variables to test existence of groups and for placement

⊞ ✓ Create ManagersGroup, if needed

⊞ ✓ Create EmployeesGroup, if needed

⊟ ✓ If Title indicates Manager, add to ManagerGroup and set rights

Conditions

✓ Condition Group 1

✓ if class name equal "User"

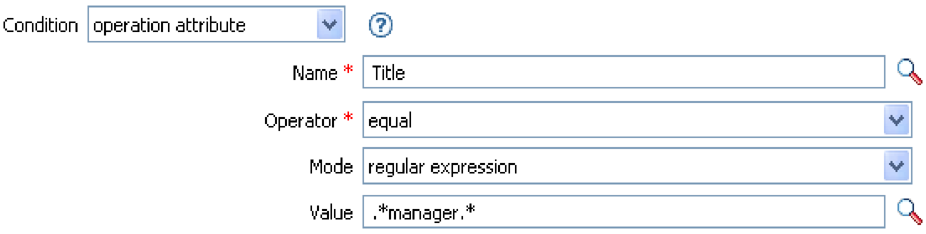
And ✓ if operation attribute 'Title' match ".*manager.*"

Actions

✓ set destination attribute value("Group Membership", Local Variable("manager-group-dn"))

✓ clone operation attribute("Group Membership", "Security Equals")

⊞ ✓ If Title does not indicate Manager, add to EmployeeGroup and set rights



Condition: operation attribute

Name *: Title

Operator *: equal

Mode: regular expression

Value: .*manager.*

The condition is checking to see if the attribute of Title is equal to .*manager*, which is a regular expression. This means that it is looking 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 and 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.

Operator

Select the condition test type.

Operator	Returns True when...
Available	There is an operation property with the specified name on the current operation.
Not Available	Available would return False.
Equal	There is a an operation property with the specified name on the current operation, and its value equals the provided content when compared using the specified comparison mode.
Not Equal	Equal would return False.
Greater Than	There is a an operation property with the specified name on the current operation, and its value is greater than the content of the condition when compared using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
Less Than	There is a an operation property with the specified name on the current operation, and its value is less than the content of the condition when compared using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

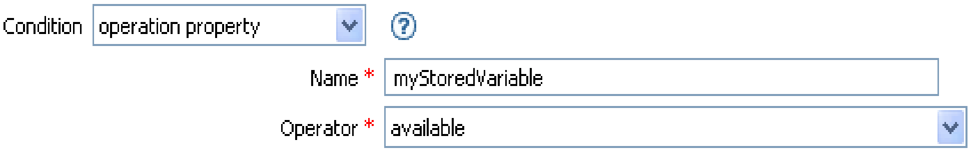
The condition has a comparison mode parameter that indicates how a comparison is done.

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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



Condition: operation property

Name *: myStoredVariable

Operator *: available

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 using the specified comparison mode.
Not Equal	Equal would return False.
Greater Than	The name of the current operation is greater than content of the condition when compared using the specified comparison mode.
Not Greater Than	Greater Than would return False.
Less Than	The name of the current operation is less than content of the condition when compared using the specified comparison mode.
Not Less Than	Less Than would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

The values are the operations that the Metadirectory engine looks for:

- add
- add-association
- check-object-password
- check-password
- delete
- get-named-password
- init-params
- instance
- modify

- modify-association
- modify-password
- move
- password
- query
- query-schema
- remove-association
- rename
- schema-def
- status
- sync

This list is not exclusive. Custom operations can be implemented by drivers and administrators.

Comparison Mode

The condition has a comparison mode parameter that indicates how a comparison is done.

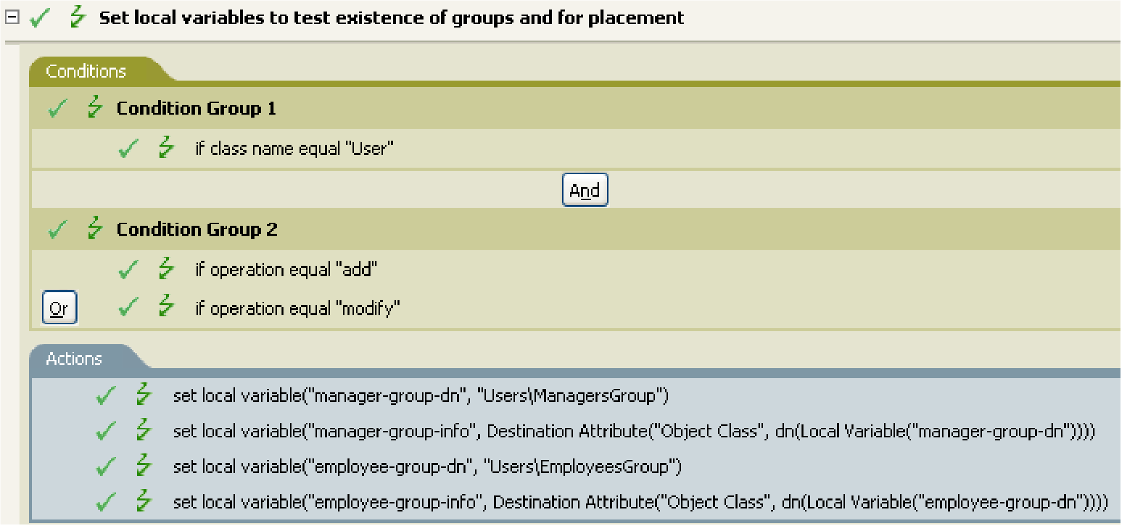
Mode	Description
Case Sensitive	Character-by-character case sensitive comparison.
Case Insensitive	Character-by-character case insensitive comparison.
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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 group, Employee or Manager, 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 003-Command-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).



Set local variables to test existence of groups and for placement

Condition Group 1

if class name equal "User"

Condition Group 2

if operation equal "add"

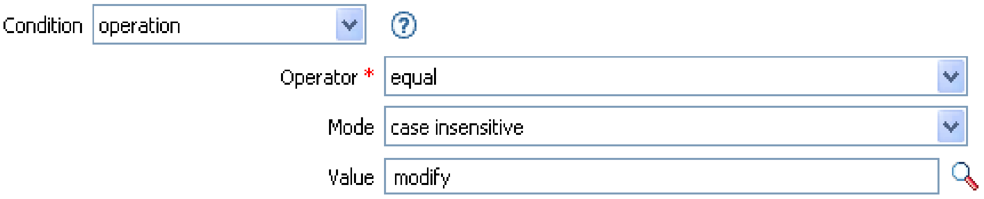
if operation equal "modify"

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"))))



Condition: operation

Operator *: equal

Mode: case insensitive

Value: modify

The condition is checking to see if an Add or Modify operation has occurred. When one of these occurs, it sets the local variables.

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.
Not Available	Available would return False.
Equal	There is a password available in the current operation, and its value equals the content of the condition when compared using the specified comparison mode.
Not Equal	Equal would return false.
Greater Than	There is a password available in the current operation, and its value is greater than the content of the condition when compared using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
Less Than	There is a password available in the current operation, and its value is less than the content of the condition when compared using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less 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.

Mode	Description
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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares using semantics appropriate to the DN format for the destination data store.
Numeric	Compares numerically.
Binary	Compares the binary information.

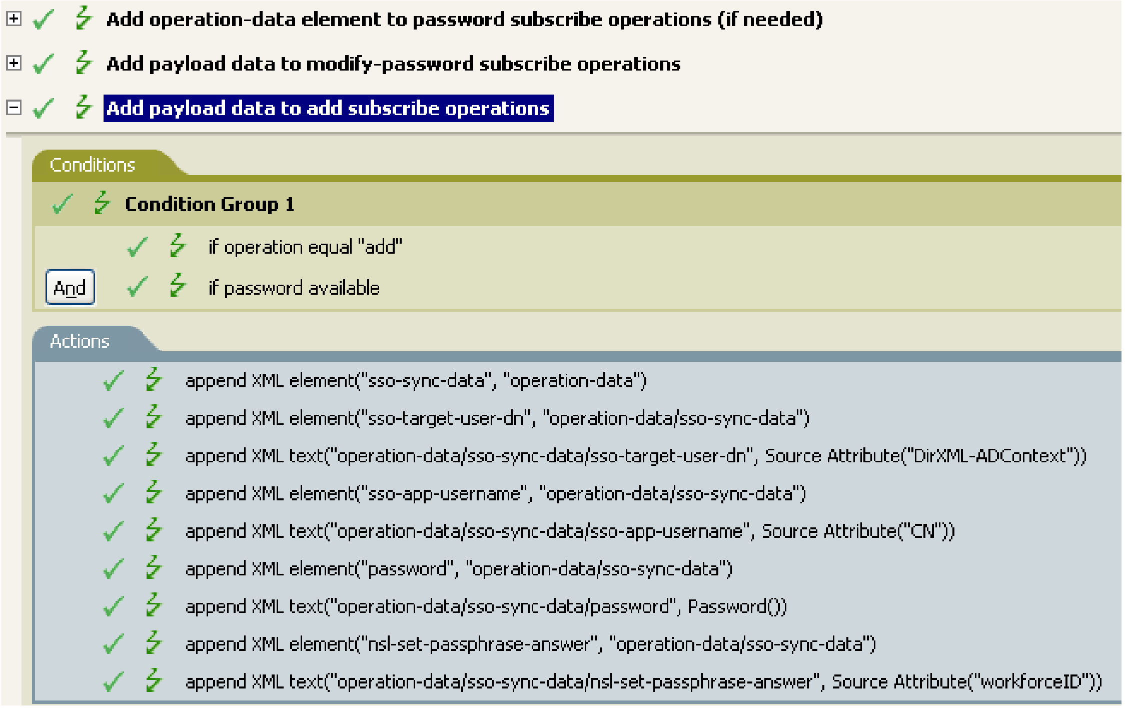
The operators that contain the comparison mode parameter are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example

If you are implementing Novell Credential Provisioning Policies Novell Credential Provisioning Policies for Identity Manager 3.5.1 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html), 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 Identity Manager media. For more information, see Example Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/credprovnlsexample.html). To view the policy in XML, see SampleSubCommandTransform.xml (../samples/SampleSubCommandTransform.xml).

The Subscriber Command Transformation policy checks if a password is available when an object is added. If the password is available, then the Novell SecureLogin and Novell 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

Conditions

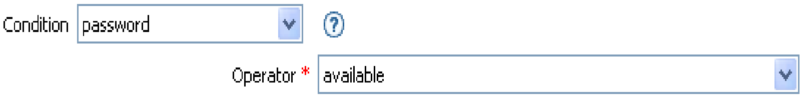
✓ Condition Group 1

✓ 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 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.

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.
Not Available	Available would return False.
Equal	There is a value available in the source data store for the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode. If the mode is structured, the content must be a set of components; otherwise, it must be text.
Not Great Than	Greater Than or Equal would return False.
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 using the specified comparison mode. If the mode is structured, the content must be a set of components; otherwise, it must be text.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

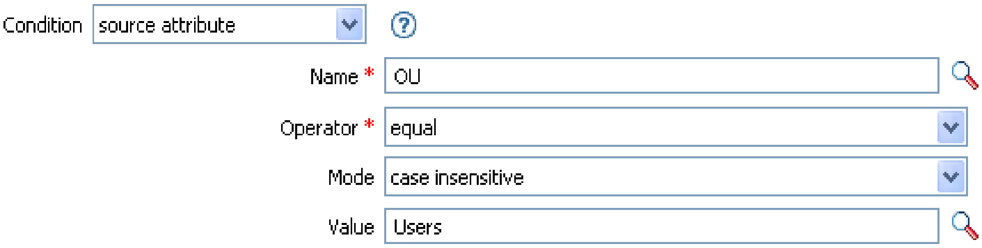
The condition has a comparison mode parameter that indicates how a comparison is done.

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Example



Condition: source attribute

Name *: OU

Operator *: equal

Mode: case insensitive

Value: Users

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.
Not Available	Available would return False.
Equal	There is a source DN available, and it equals the content of the specified value in-container.
Not Equal	Equal would return False.
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 using semantics appropriate to the DN format of the source data store.
Not in Container	In Container would return False.
In Subtree	There is a source DN available, and it represents an object in the subtree identified by the specified value.
Not in subtree	In Subtree would return False.

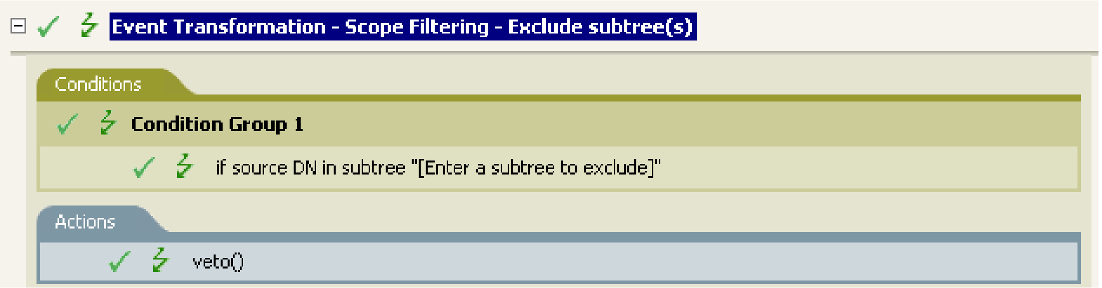
Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- In Container
- Not in Container
- In Subtree
- 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 Identity Manager. For more information, see Event Transformation - Scope Filtering - Exclude Subtrees (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prfilterexcludesubtree.html#prfilterexcludesubtree). To view the policy in XML, see predef_transformation_exclude_subtress.xml (../samples/predef_transformation_exclude_subtrees.xml).



Event Transformation - Scope Filtering - Exclude subtree(s)

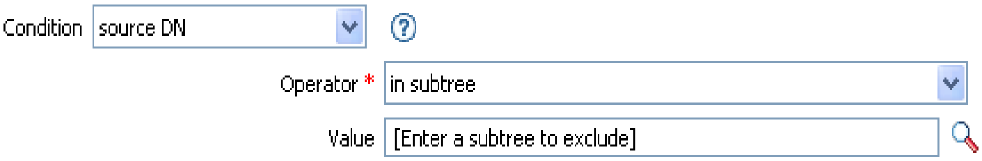
Conditions

Condition Group 1

if source DN in subtree "[Enter a subtree to exclude]"

Actions

veto()



Condition: source DN

Operator *: in subtree

Value: [Enter a subtree to exclude]

The condition is checking to see if the source DN is in the Users container. If the object is coming 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.
Not available	Available would return False.
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 using the specified comparison mode.
Not Equal	Equal would return False.
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 using the specified comparison mode.
Not Greater Than	Greater Than or Equal would return False.
Less Than	The association value specified by the current operation is less than the content of the condition when compared using the specified comparison mode.
Not Less Than	Less Than or Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. Each value supports variable expansion. For more information, see Variable Expansion (page 211). The operators that contain the value field are:

- Equal
- Not Equal
- Greater Than
- Not Greater Than
- Less Than
- Not Less Than

Comparison Mode

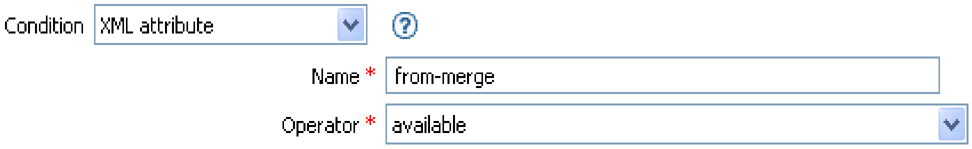
The condition has a comparison mode parameter that indicates how a comparison is done.

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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



Condition XML attribute

Name * from-merge

Operator * available

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.

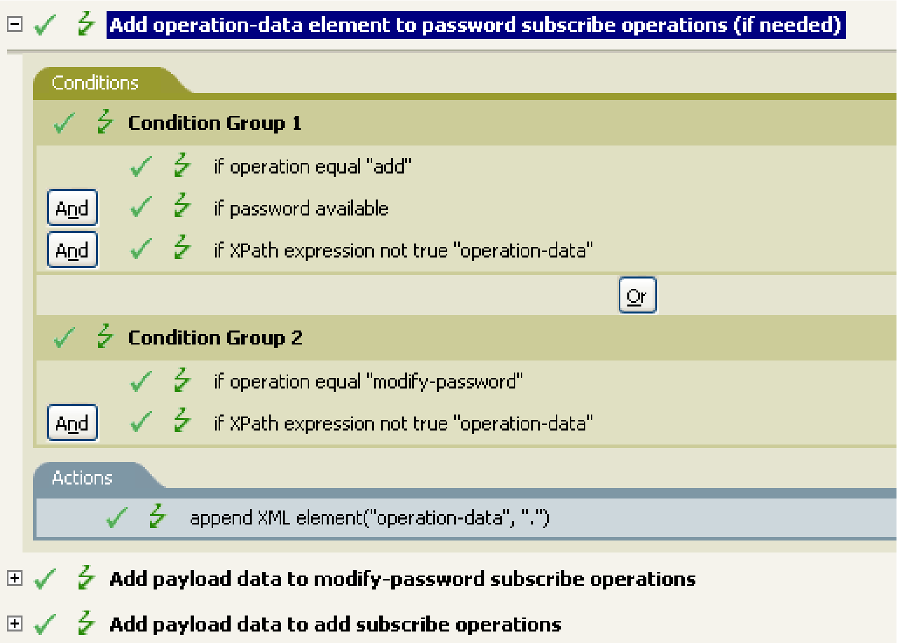
Remarks

For more information on using XPath expression with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example

If you are implementing Novell 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 Identity Manager media. For more information, see Example Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/credprovnlsexample.html). To view the policy in XML, see SampleSubCommandTransform.xml (../samples/SampleSubCommandTransform.xml).

The sample Credential Provisioning policy is checking each Add operation to see if there is operation data associated with the Add. If there is no operation data, the Novell SecureLogin and Novell SecretStore credentials are provisioned.



✓ Add operation-data element to password subscribe operations (if needed)

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"

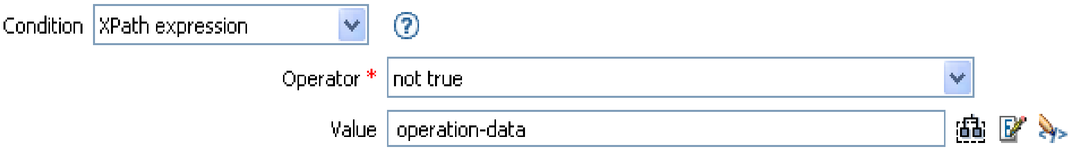
And ✓ if XPath expression not true "operation-data"

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

Variable Expansion

Allows for the use of dynamic variables in the condition

Remark

Many conditions 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 or global variable with the given name. $<variable-name>$ must be legal variable name. For information on what is 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, it should be escaped with an additional $ (for example, You owe me $$100.00).

Actions

13

Actions are performed when conditions of the enclosing rule are met. Some actions have a *Mode* field. The mode is not honored at run time if the context in which the policy is running is incompatible with the selected mode.

This section contains detailed information about all actions that are available through using the Policy Builder interface.

- “Send Email” on page 257
- “Send Email from Template” on page 259
- “Set Default Attribute Value” on page 261
- “Set Destination Attribute Value” on page 263
- “Set Destination Password” on page 265
- “Set Local Variable” on page 266
- “Set Operation Association” on page 268
- “Set Operation Class Name” on page 269
- “Set Operation Destination DN” on page 270
- “Set Operation Property” on page 271
- “Set Operation Source DN” on page 272
- “Set Operation Template DN” on page 273
- “Set Source Attribute Value” on page 274
- “Set Source Password” on page 276
- “Set SSO Credential” on page 277
- “Set SSO Passphrase” on page 278
- “Set XML Attribute” on page 279
- “Status” on page 280
- “Start Workflow” on page 281
- “Strip Operation Attribute” on page 283
- “Strip XPath” on page 284
- “Trace Message” on page 285
- “Veto” on page 287
- “Veto If Operation Attribute Not Available” on page 288
- “While” on page 289
- “Variable Expansion” on page 290

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.

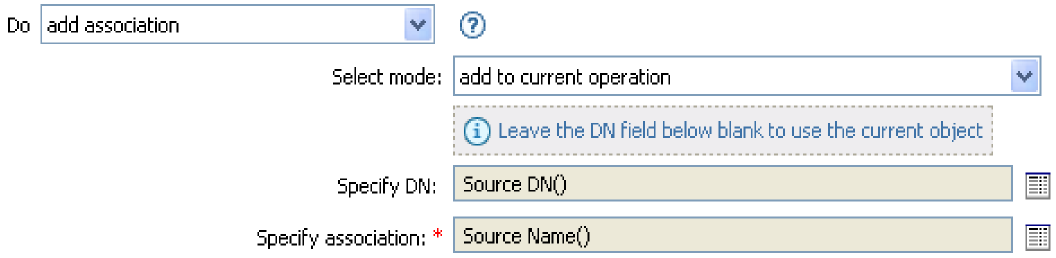
DN

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



Do add association

Select mode: add to current operation

Leave the DN field below blank to use the current object

Specify DN: Source DN()

Specify association: * Source Name()

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 Expansion (page 290).

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 Expansion (page 290).

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

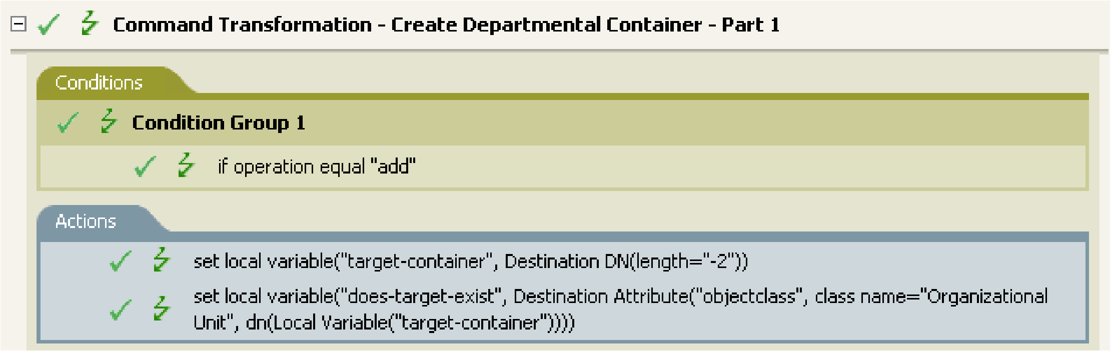
Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

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 Identity Manager. For more information, see Command Transformation - Create Departmental Container - Part 1 and Part2 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeptcontainer.html#prdeptcontainer). 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

Conditions

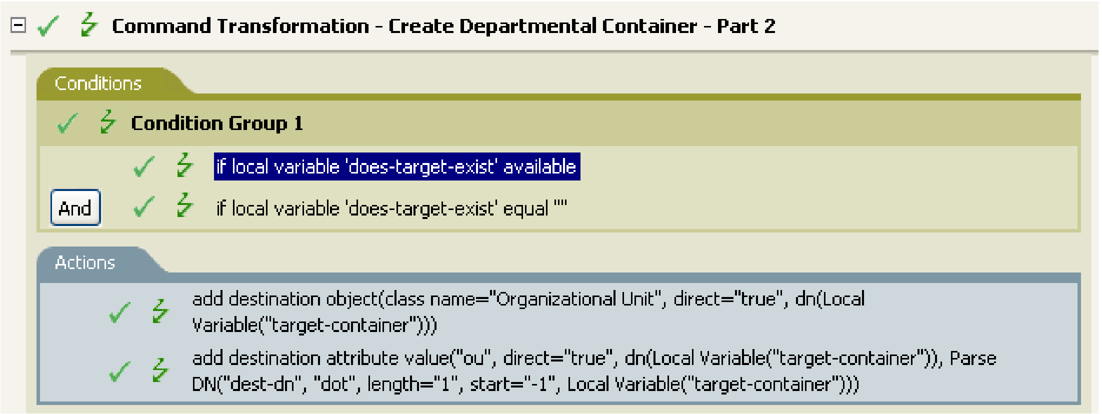
Condition Group 1

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

Conditions

Condition Group 1

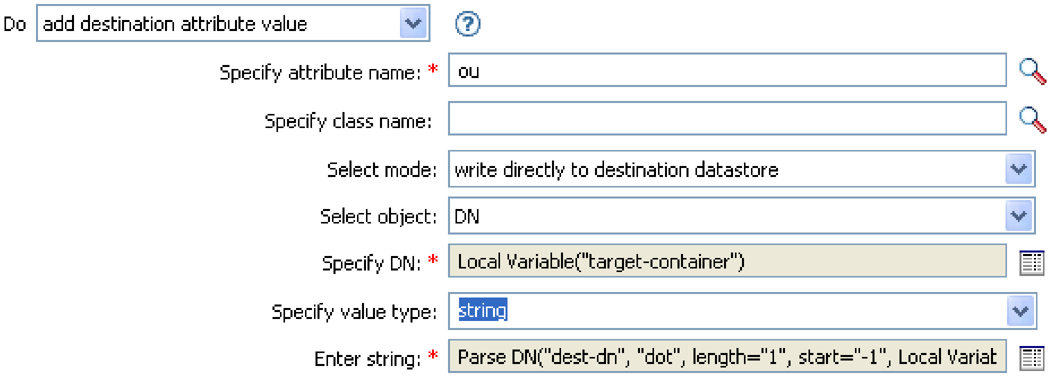
if local variable 'does-target-exist' available

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: * ou

Specify class name:

Select mode: write directly to destination datastore

Select object: DN

Specify DN: * Local Variable("target-container")

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 Expansion (page 290).

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

DN

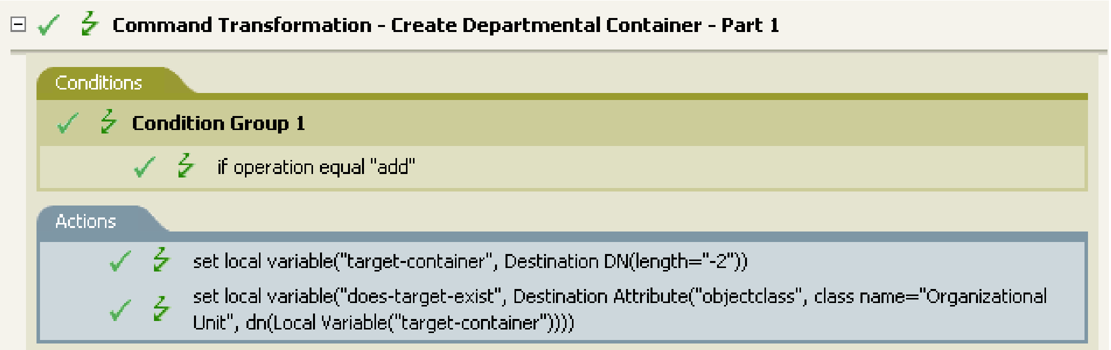
Specify the DN of the object to be created.

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 Identity Manager. For more information, see Command Transformation - Create Departmental Container - Part 1 and Part2 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeptcontainer.html#prdeptcontainer) 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).



⊟ ✓ Command Transformation - Create Departmental Container - Part 1

Conditions

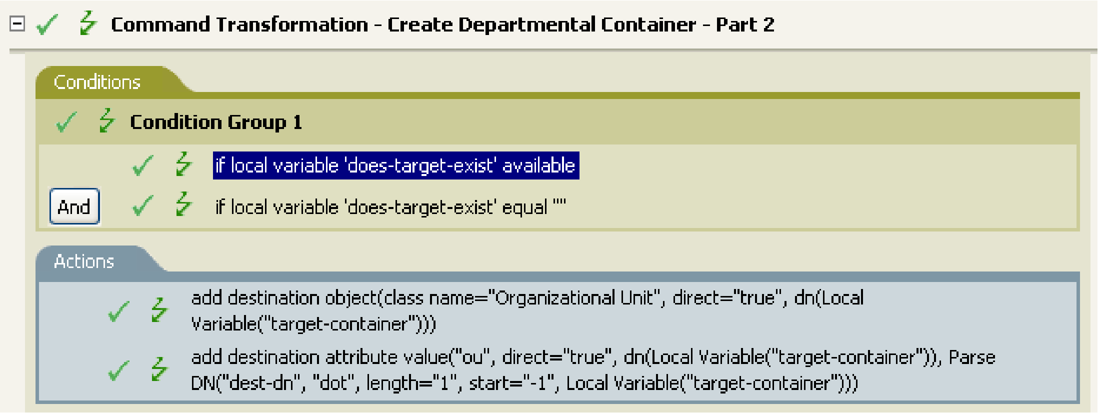
✓ Condition Group 1

✓ 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

Conditions

Condition Group 1

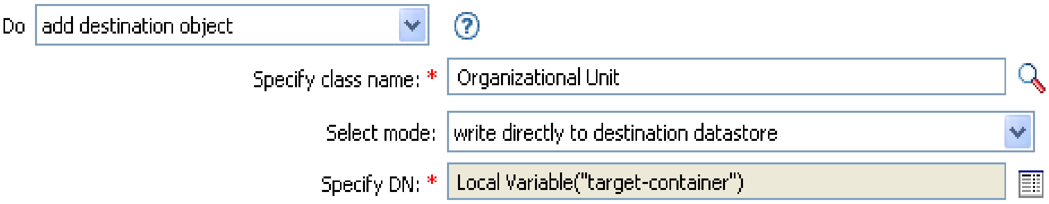
if local variable 'does-target-exist' available

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 object

Specify class name: * Organizational Unit

Select mode: write directly to destination datastore

Specify DN: * Local Variable("target-container")

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 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 Expansion (page 290).

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 Expansion (page 290).

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.

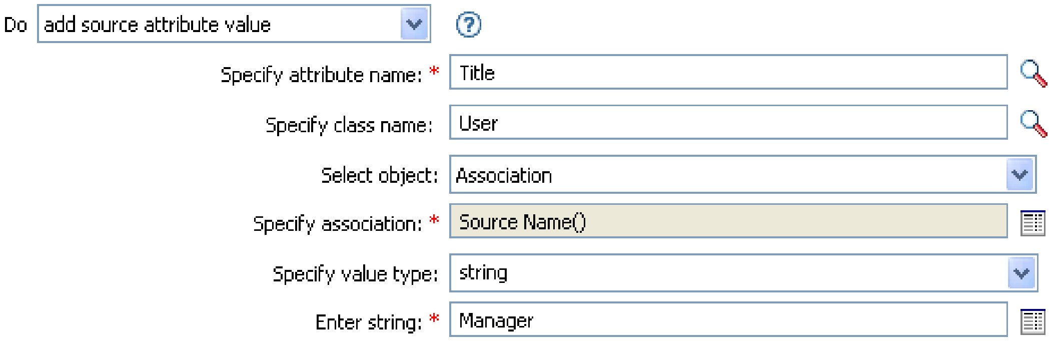
Value Type

Select the syntax of the attribute value to be added. The options are string, counter, dn, int, interval, octet, state, structured, teleNumber, or time.

Value

Specify the attribute value to be added.

Example



Do: add source attribute value

Specify attribute name: * Title

Specify class name: User

Select object: Association

Specify association: * Source Name()

Specify value type: string

Enter string: * Manager

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

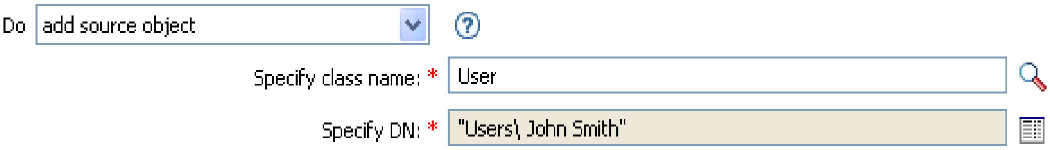
Class Name

Specify the class name of the object to be added. Supports variable expansion. For more information, see Variable Expansion (page 290).

DN

Specify the DN of the object to be added.

Example



Do add source object

Specify class name: * User

Specify DN: * "Users\ John Smith"

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 Expansion (page 290).

XPath Expression

Specify an XPath 1.0 expression that returns a node set containing the elements to which the new elements should be appended.

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.

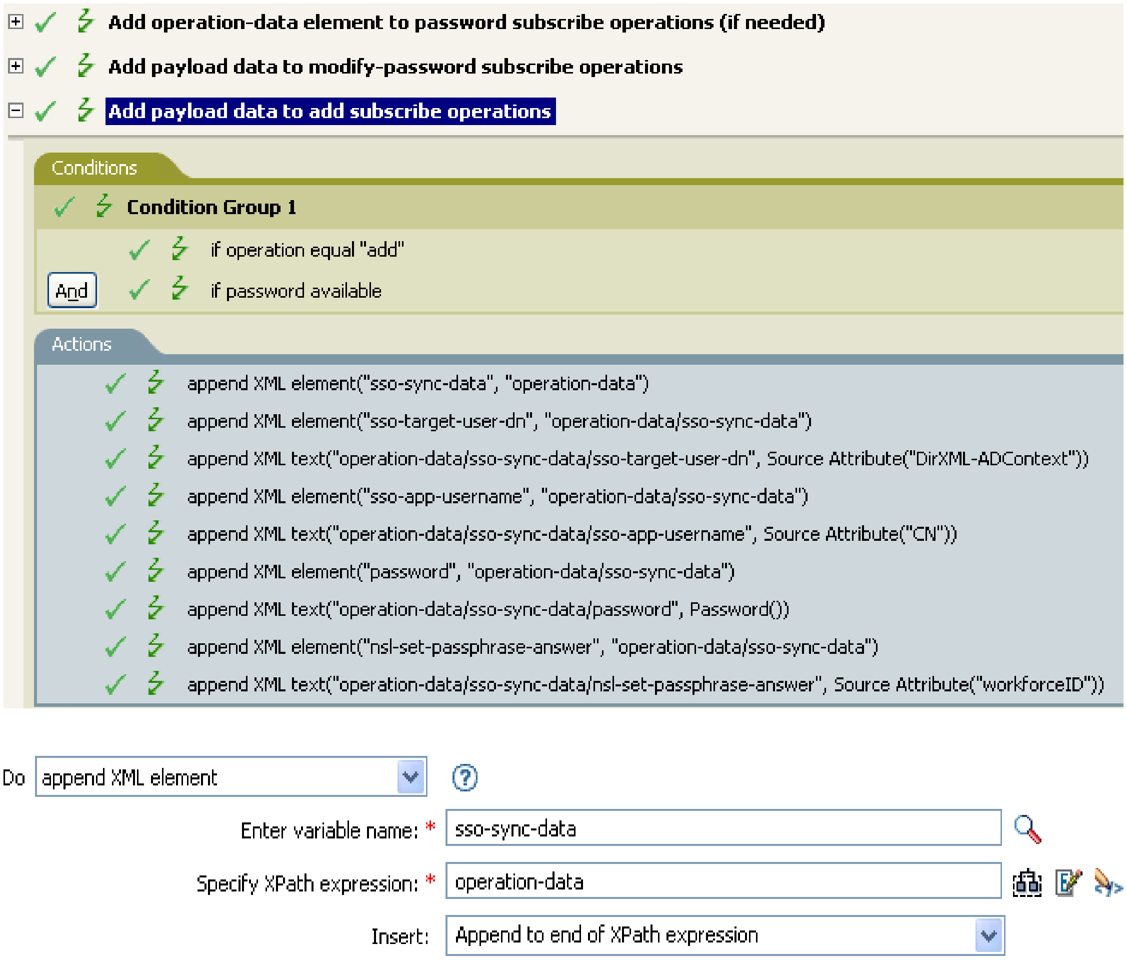
Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example

If you are implementing Novell 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 Identity Manager media. For more information, see Example Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/credprovnlsexample.html). To view the policy in XML, see SampleSubCommandTransform.xml (../samples/SampleSubCommandTransform.xml).

The sample file uses the *append XML element* action to add the Novell® SecureLogin or Novell 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**

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 variable 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 node set selected by before 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.

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.

String

Specify the text to be appended.

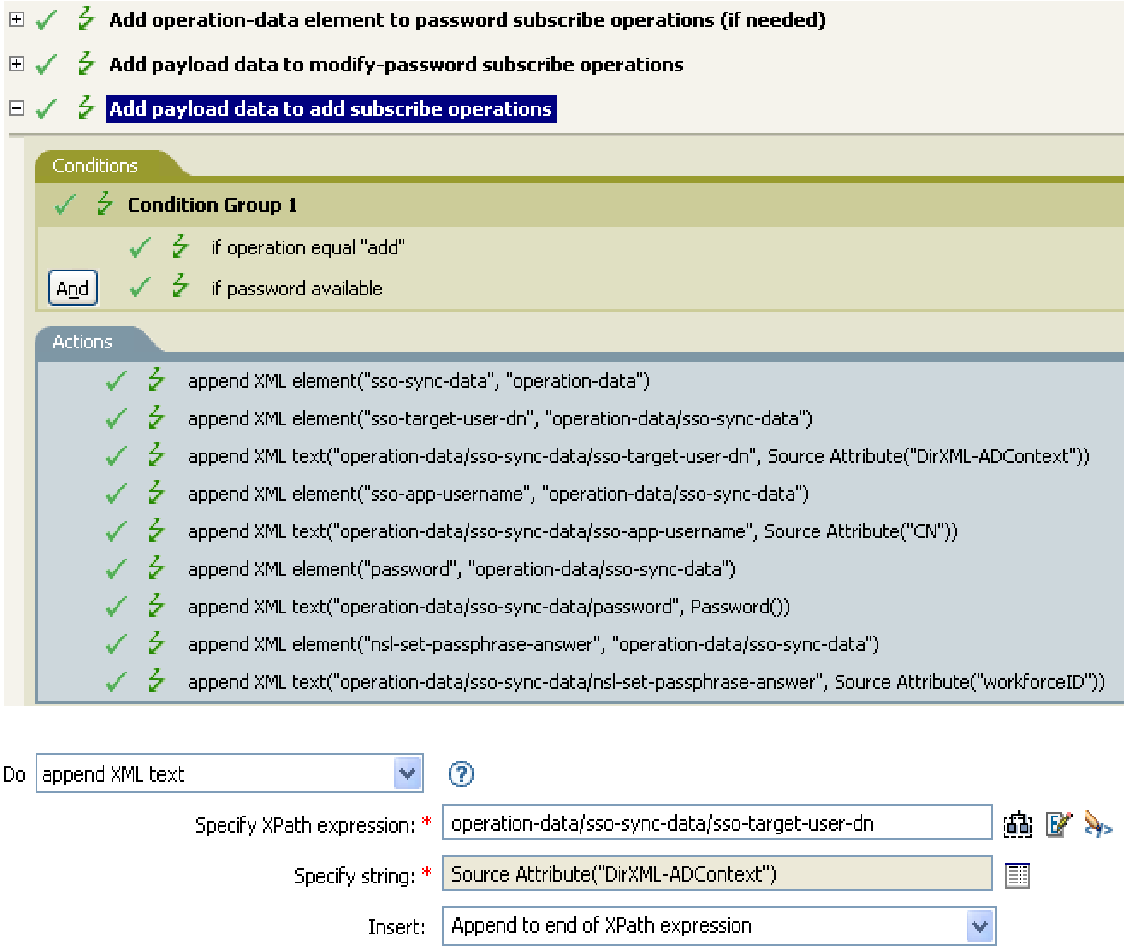
Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example

If you are implementing Novell 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 Identity Manager media. For more information, see Example Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/credprovnlsexample.html). To view the policy in XML, see SampleSubCommandTransform.xml (../samples/SampleSubCommandTransform.xml).

The example is using the *append XML text* action to find the Novell SecureLogin or Novell 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

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

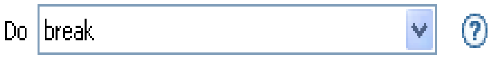
Specify string: * Source Attribute("DirXML-ADContext")

Insert: Append to end of XPath expression

Break

Ends processing of the current operation by the current policy.

Example



Do break

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 Expansion (page 290).

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 Expansion (page 290).

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

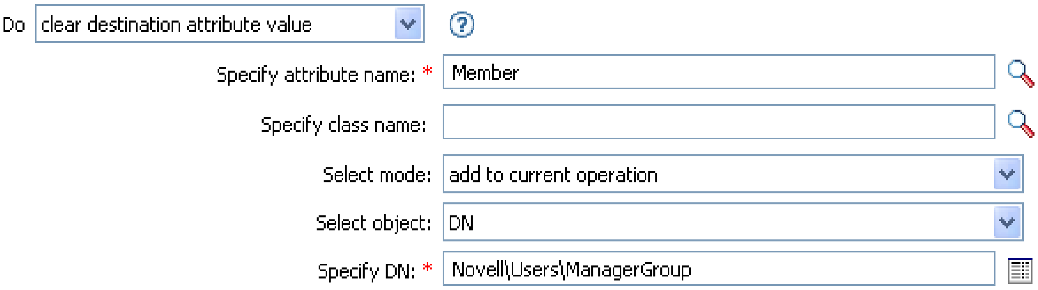
Object

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 the current object as the target object.

Example



Do clear destination attribute value

Specify attribute name: * Member

Specify class name:

Select mode: add to current operation

Select object: DN

Specify DN: * Novell\Users\ManagerGroup

Clear Operation Property

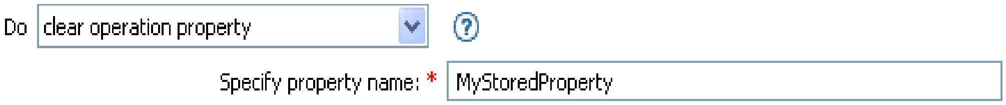
Clears any operation property with the provided name from the current operation. The operation property is the XML attribute attached to an `<operation-data>` element by a policy. An XML attribute is a name/value pair associated with an element in the XDS document.

Fields

Property Name

Specify the name of the operation property to clear. Supports variable expansion. For more information, see Variable Expansion (page 290).

Example



Do clear operation property

Specify property name: * MyStoredProperty

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 Expansion (page 290).

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 mapping purposes if the object is other than current object. Supports variable expansion. For more information, see Variable Expansion (page 290).

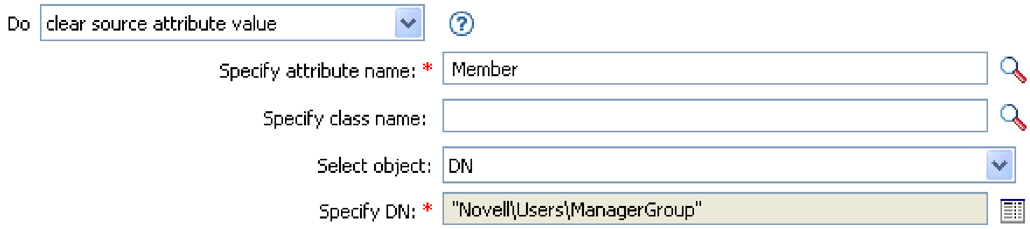
Object

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



Do clear source attribute value

Specify attribute name: * Member

Specify class name:

Select object: DN

Specify DN: * "Novell\Users\ManagerGroup"

Clear SSO Credential

Clears the Single Sign On credential so objects can be deprovisioned. Additional information about the credential to be cleared can be entered 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 Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Target User DN

Specify the DN of the target users.

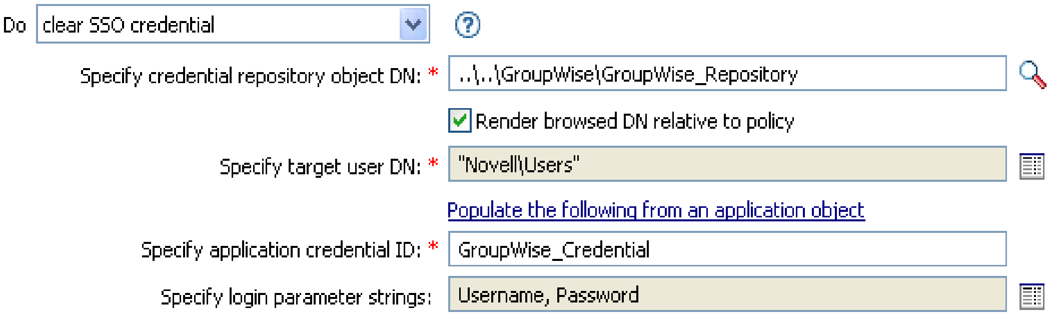
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Example



Do: clear SSO credential

Specify credential repository object DN: * ..\..\GroupWise\GroupWise_Repository

☑ Render browsed DN relative to policy

Specify target user DN: * "Novell\Users"

Populate the following from an application object

Specify application credential ID: * GroupWise_Credential

Specify login parameter strings: Username, Password

Clone By XPath Expression

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 node set previously selected 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.

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.

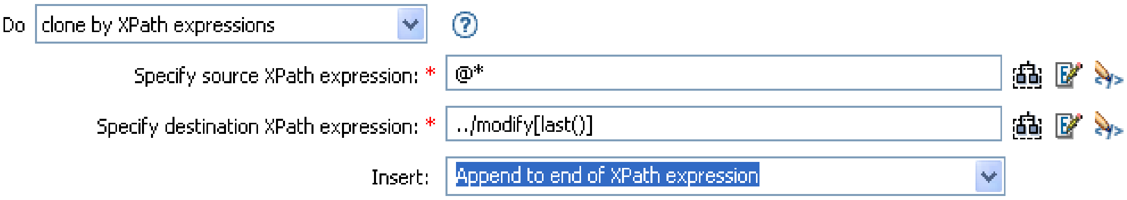
Insert

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.

Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example



Do: clone by XPath expressions

Specify source XPath expression: * @*

Specify destination XPath expression: * ../modify[last()]

Insert: Append to end of XPath expression

Clone Operation Attribute

Copies all occurrences of an attribute within the current operation to a different attribute within the current operation.

Fields

Source Name

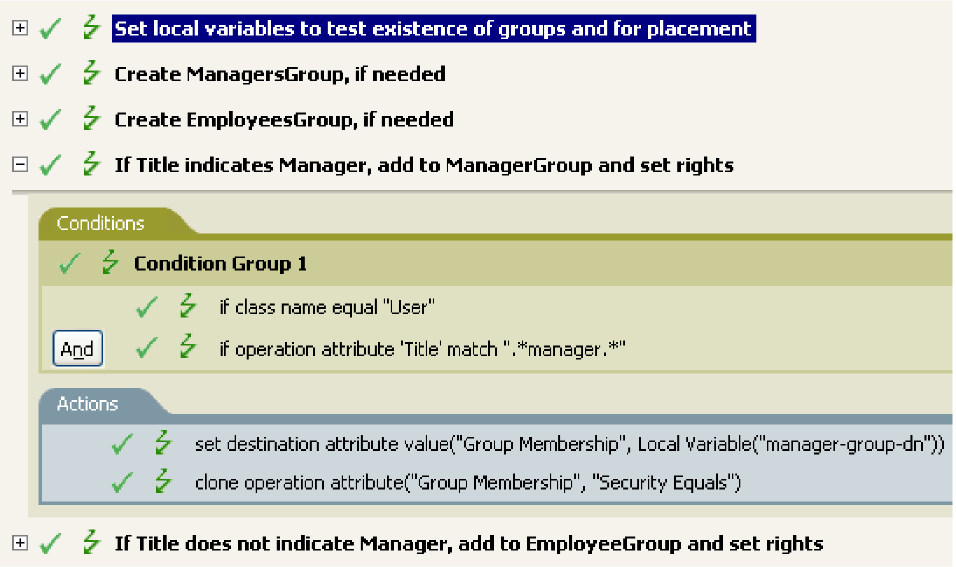
Specify the name of the attribute to be copied from. Supports variable expansion. For more information, see Variable Expansion (page 290).

Destination Name

Specify the name of the attribute to be copied to. Supports variable expansion. For more information, see Variable Expansion (page 290).

Example

The example adds a User object to the appropriate group, Employee or Manager, 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To see the policy in XML, see 003-AddCreateGroups.xml (../samples/003-AddCreateGroups.xml).



⊞ ✓ Set local variables to test existence of groups and for placement

⊞ ✓ Create ManagersGroup, if needed

⊞ ✓ Create EmployeesGroup, if needed

⊟ ✓ If Title indicates Manager, add to ManagerGroup and set rights

Conditions

✓ Condition Group 1

✓ if class name equal "User"

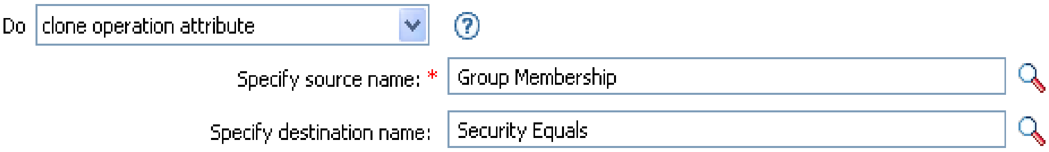
And ✓ if operation attribute 'Title' match ".*manager.*"

Actions

✓ set destination attribute value("Group Membership", Local Variable("manager-group-dn"))

✓ clone operation attribute("Group Membership", "Security Equals")

⊞ ✓ If Title does not indicate Manager, add to EmployeeGroup and set rights



Do clone operation attribute

Specify source name: * Group Membership

Specify destination name: Security Equals

The Clone Operation Attribute is taking the information from the Group Membership attribute and adding that to the Security Equals attribute so the values are the same.

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.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

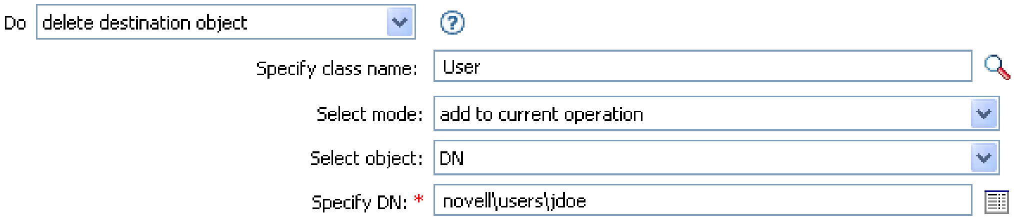
Object

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



Do: delete destination object

Specify class name: User

Select mode: add to current operation

Select object: DN

Specify DN: * novell\users\jdoe

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.

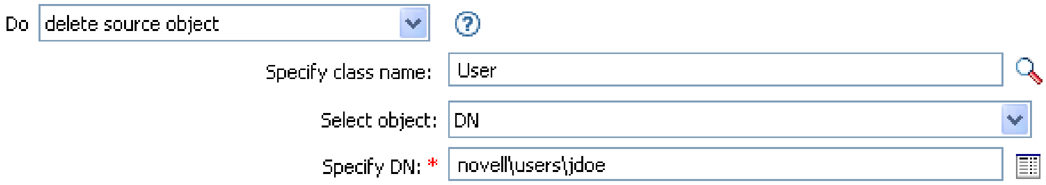
Object

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



Do: delete source object

Specify class name: User

Select object: DN

Specify DN: * novell\users\jdoe

Find Matching Object

Finds a match for the current object in the destination data store.

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.

Remarks

Find Matching Object is only valid when the current operation is an add.

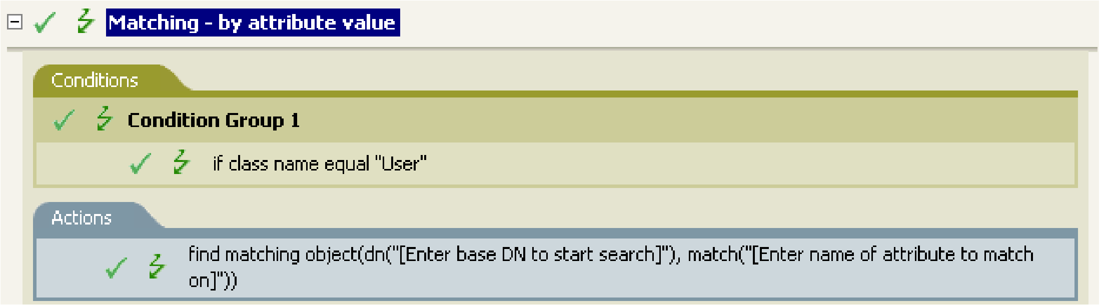
The DN argument is required when scope is "entry," and is optional otherwise. At least one match attribute is required when scope is "subtree" or "subordinates."

The results are undefined if 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 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 &#xFFFC;. If multiple results are returned, then the destination DN of the current operation is set to the single character &#xFFFD;.

Example

The example matches on Users 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 Identity Manager. For more information, see Matching - By Attribute Value (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prmatchattrvalue.html#prmatchattrvalue). To see the policy in XML, see predef_match_by_attribute.xml (../samples/predef_match_by_attribute.xml).



Matching - by attribute value

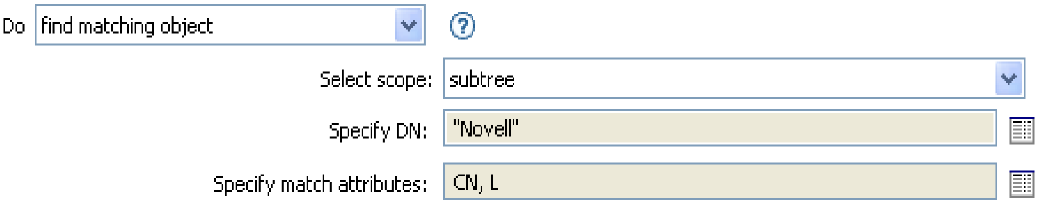
Conditions

Condition Group 1

if class name equal "User"

Actions

find matching object(dn("[Enter base DN to start search]"), match("[Enter name of attribute to match on]"))



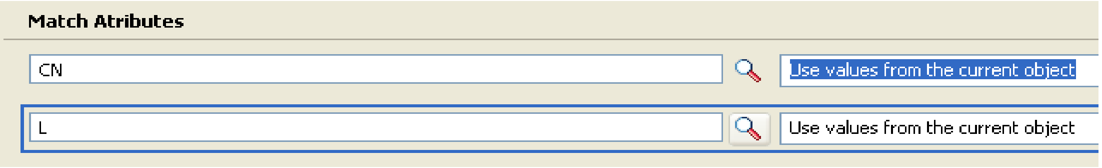
Do: find matching object

Select scope: subtree

Specify DN: "Novell"

Specify match attributes: CN, L

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.



Match Atributes

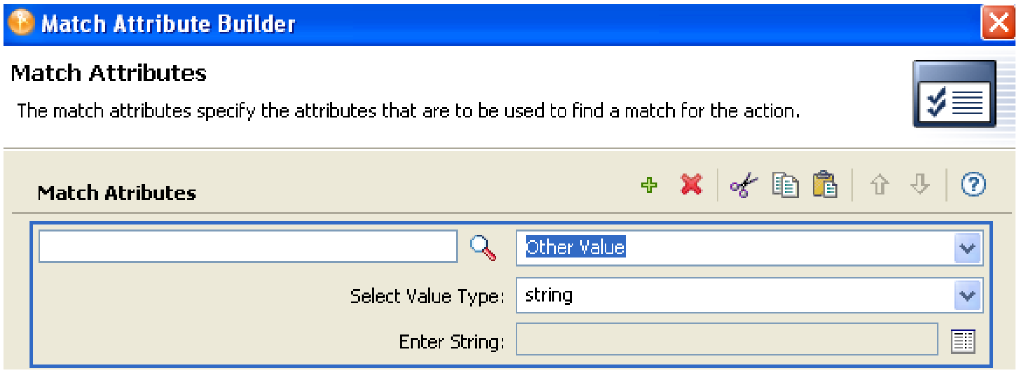
CN	Use values from the current object
L	Use values from the current object

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 the *Other Value*:

1 Launch the Match Attribute Builder, then select *Other Value*.



Match Attribute Builder
Match Attributes
The match attributes specify the attributes that are to be used to find a match for the action.
Match Atributes

Other Value

Select Value Type: string

Enter String:

2 Select the desired value type.

3 Specify the value, then click *OK*.

For Each

Repeats a set of actions for each node in a node set.

Fields

Node Set

Specify the node set.

Action

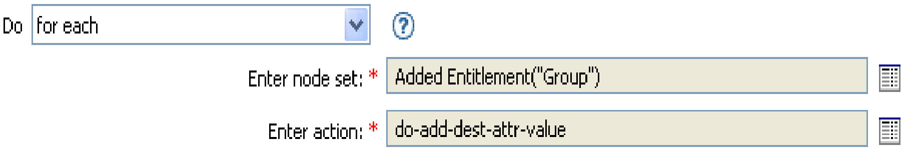
Specify the actions to perform on each node in the node set.

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

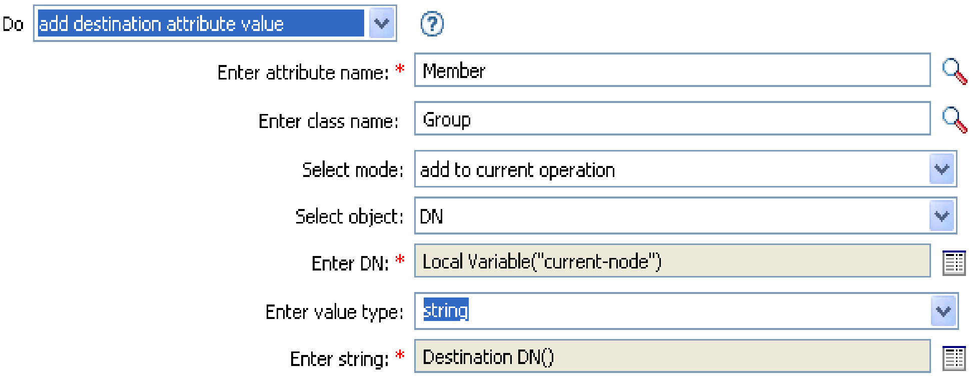


Do for each

Enter node set: * Added Entitlement("Group")

Enter action: * do-add-dest-attr-value

The following is an example of the Argument Actions Builder, used to provide the action argument:



Do add destination attribute value

Enter attribute name: * Member

Enter class name: Group

Select mode: add to current operation

Select object: DN

Enter DN: * Local Variable("current-node")

Enter value type: string

Enter string: * Destination DN()

Generate Event

Sends a user-defined event to Novell Audit or Sentinel.

Fields

ID

ID of the event. The provided value must result in an integer in the range of 1000-1999 when parsed using the parseInt method of java.lang.Integer. Supports variable expansion. For more information, see Variable Expansion (page 290).

Level

Level of the event.

Level	Description
log-emergency	Events that cause the Metadirectory engine or driver to shut down.
log-alert	Events that require immediate attention.
log-critical	Events that can cause parts of the Metadirectory engine or driver to malfunction.
log-error	Events describing errors that can be handled by the Metadirectory engine or driver.
log-warning	Negative events not representing a problem.
log-notice	Events (positive or negative) that an administrator can use to understand or improve use and operation.
log-info	Positive events of any importance.
log-debug	Events of relevance for support or engineers to debug the operation of the Metadirectory engine or driver.

Strings

Specify user-defined string, integer, and binary values to include with the event. These values are provided using the Named String Builder.

Tag	Description
target	The object being acted upon.
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
subTarget	The subcomponent of the target being acted upon.
text1	Text entered here is stored in the text1 event field.

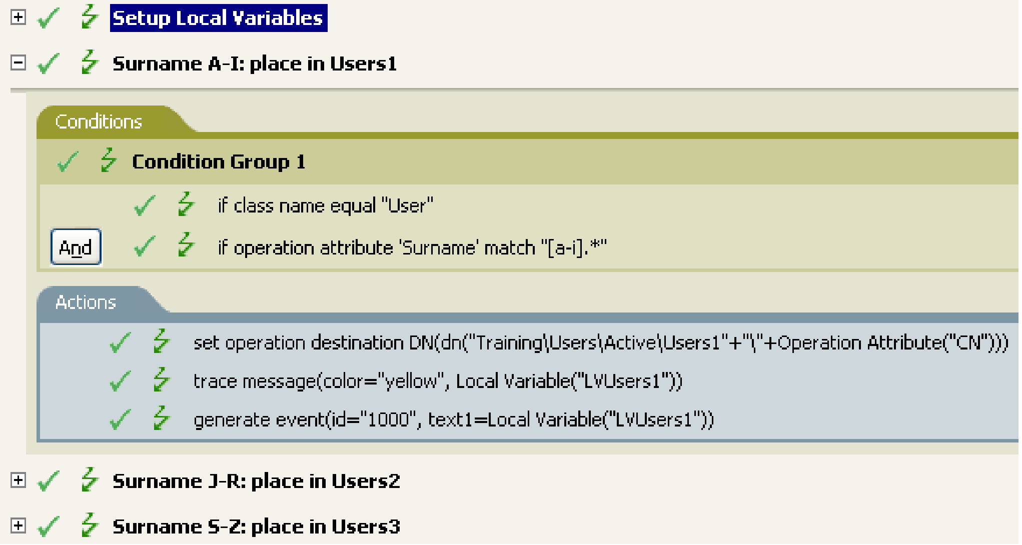
Tag	Description
text2	Text entered here is stored in the text2 event field.
text3	Text entered here is stored in the text3 event field.
value	Any number entered here is stored in the value event field.
value3	Any number entered here is stored in the value3 event field.
data	Data entered here is stored in the blob event field.

Remarks

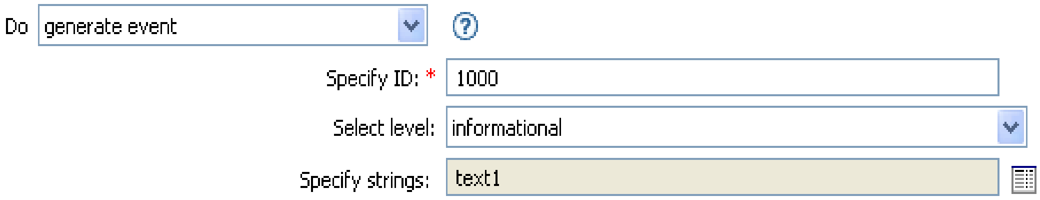
The Novell Audit or Sentinel event structure contains a target, a subTarget, three strings (text1, text2, text3), two integers (value, value3), and a generic field (data). The text fields are limited to 256 bytes, and the data field can contain up to 3 KB of information, unless a larger data field is 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 Novell Audit or Sentinel event. The Generate Event action is used to send Novell Audit or Sentinel an event. The policy name is Policy to Place by Surname and is available for download from the Novell Support Web site. For more information Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). 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
    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
```



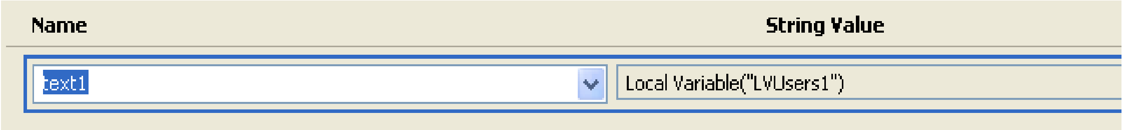
Do generate event

Specify ID: * 1000

Select level: informational

Specify strings: text1

The following is an example of the Named String Builder, used to provide the strings argument.



Name	String Value
text1	Local Variable("LVUsers1")

Generate Event is creating an event with the ID 1000 and displaying the text that is generated by the local variable of LVUser1. The local variable LVUser1 is the string of User:Operation Attribute "cn" +" added to the "+"Training\Users\Active\Users1"+" container". The event reads User:jsmith added to the Training\Users\Active\Users1 container.

If

Conditionally performs a set of actions.

Fields

If Conditions

Specify the desired condition.

Then Perform Actions

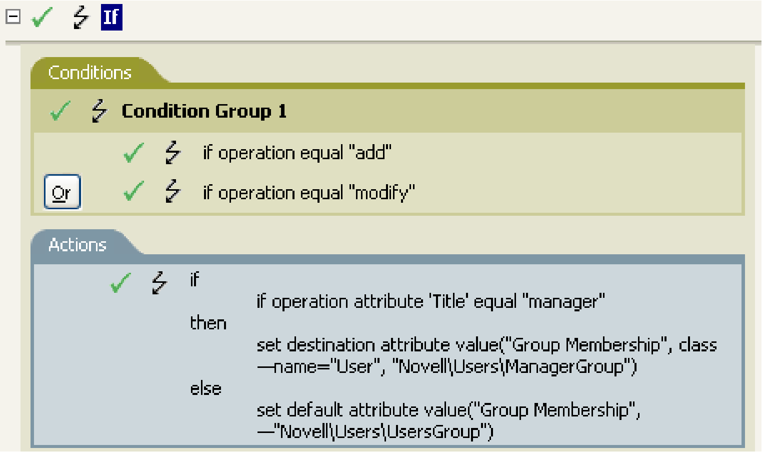
Specify the desired actions, if the conditions are True.

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).



If

Conditions

✓ Condition Group 1

✓ if operation equal "add"

Or ✓ if operation equal "modify"

Actions

✓ if

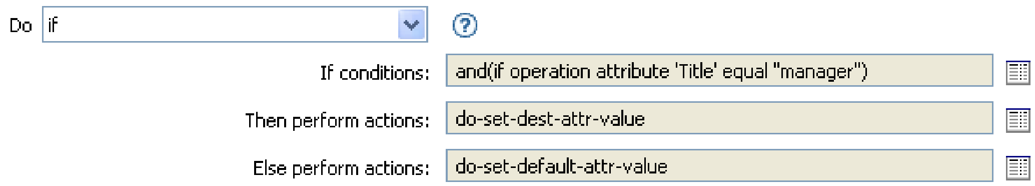
if operation attribute 'Title' equal "manager"

then

set destination attribute value("Group Membership", class —name="User", "Novell\Users\ManagerGroup")

else

set default attribute value("Group Membership", —"Novell\Users\UsersGroup")



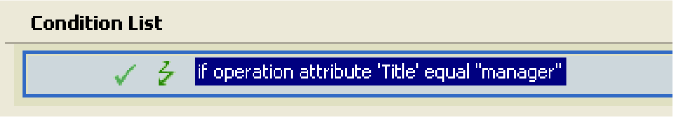
Do: if

If conditions: and(if operation attribute 'Title' equal "manager")

Then perform actions: do-set-dest-attr-value

Else perform actions: do-set-default-attr-value

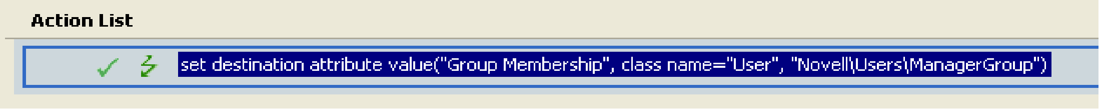
When you create the if action, you must add a condition and one action. In this example, there are two separate actions. The condition is if a user object has the title of manager.



Condition List

✓ if operation attribute 'Title' equal "manager"

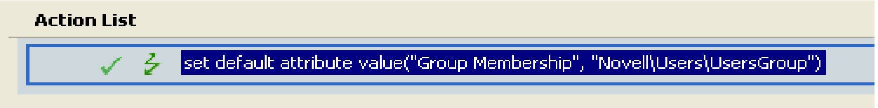
The action is to add the user object to the ManagerGroup group.



Action List

✓ set destination attribute value("Group Membership", class name="User", "Novell\Users\ManagerGroup")

If the title does not equal manager, the user object is placed in the UsersGroup group.



Action List

✓ set default attribute value("Group Membership", "Novell\Users\UsersGroup")

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

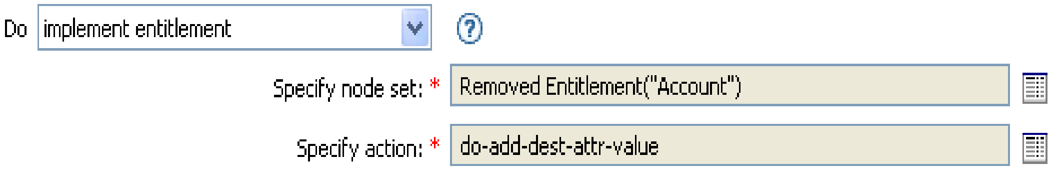
Node Set

Node set containing the entitlement being implemented by the specified actions.

Action

Actions that implement the specified entitlements.

Example

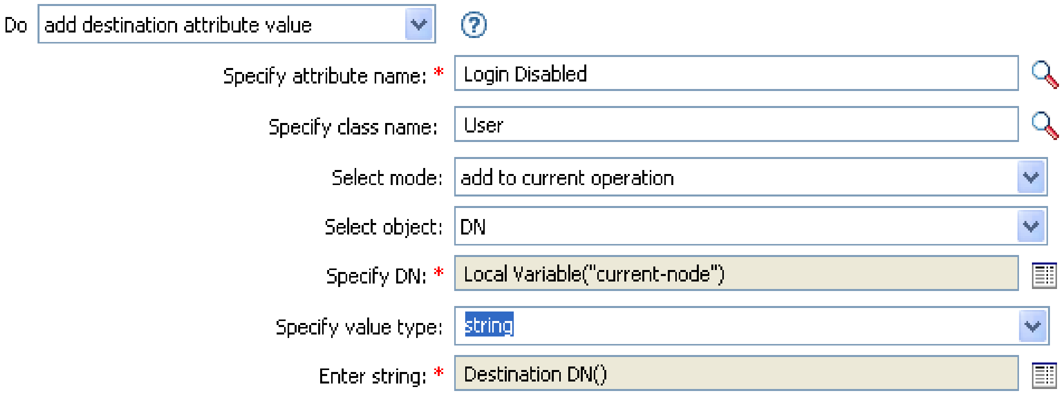


Do implement entitlement

Specify node set: * Removed Entitlement("Account")

Specify action: * do-add-dest-attr-value

The following is an example of the Argument Actions Builder, used to provide the action argument:



Do add destination attribute value

Specify attribute name: * Login Disabled

Specify class name: User

Select mode: add to current operation

Select object: DN

Specify DN: * Local Variable("current-node")

Specify value type: string

Enter string: * Destination DN()

Move Destination Object

Moves an object into the destination data store.

Fields

Class Name

(Optional) Specify the class name of the object to move into the destination data store.

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

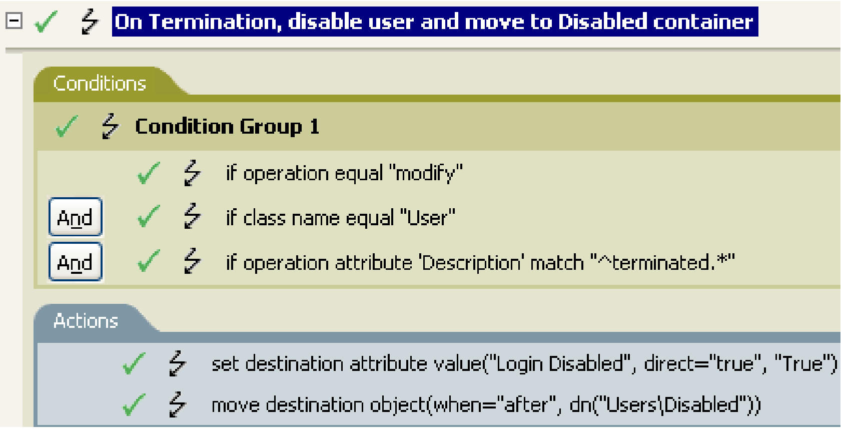
Select the container to receive the object. This container is specified by a DN or an association.

DN or Association

Specify whether the DN or association of the container is used.

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view this policy in XML, see 005-Command-DisableMoveOnTermination (../samples/005-Command-DisableMoveOnTermination.xml).



On Termination, disable user and move to Disabled container

Condition Group 1

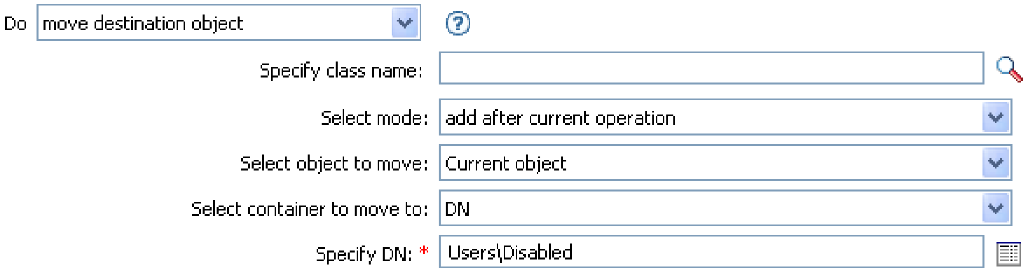
if operation equal "modify"

if class name equal "User"

if operation attribute 'Description' match "^terminated.*"

set destination attribute value("Login Disabled", direct="true", "True")

move destination object(when="after", dn("Users\Disabled"))



Do: move destination object

Specify class name:

Select mode: add after current operation

Select object to move: Current object

Select container to move to: DN

Specify DN: * Users\Disabled

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

Moves an object into the source data store.

Fields

Class Name

(Optional) Specify the class name of the object to move into the source data store.

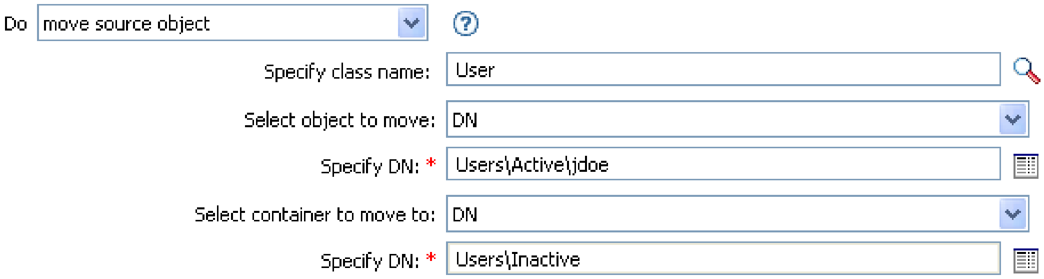
Object to Move

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.

Example



Do: move source object

Specify class name: User

Select object to move: DN

Specify DN: * Users\Active\jdoe

Select container to move to: DN

Specify DN: * Users\Inactive

Reformat Operation Attribute Value

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 Expansion (page 290).

Value Type

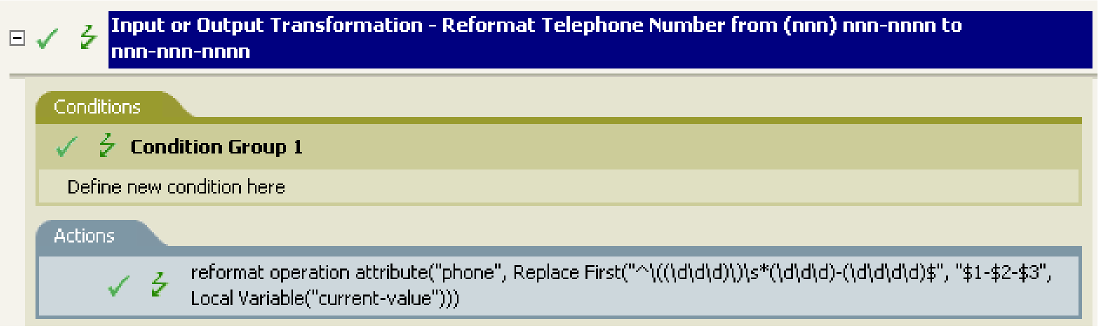
Specify the syntax of the new attribute value.

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 Identity Manager. For more information, see Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prreformattel1.html#prreformattel1). To view the policy in XML, see predef_transformation_reformat_telephone1.xml (../samples/predef_transformation_reformat_telephone1.xml).



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

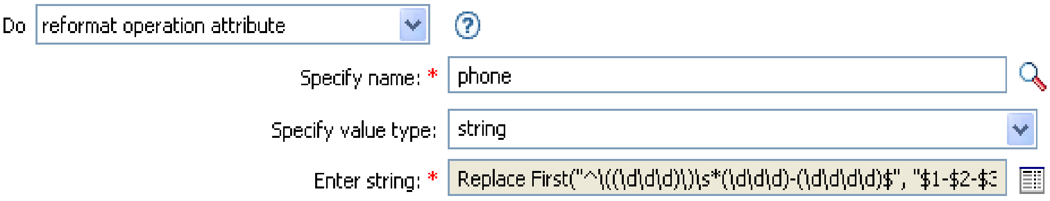
Conditions

Condition Group 1

Define new condition here

Actions

```
reformat operation attribute("phone", Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3", Local Variable("current-value")))
```



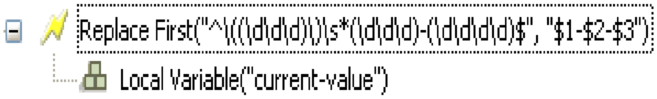
Do: reformat operation attribute

Specify name: * phone

Specify value type: string

Enter string: * `Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3`

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

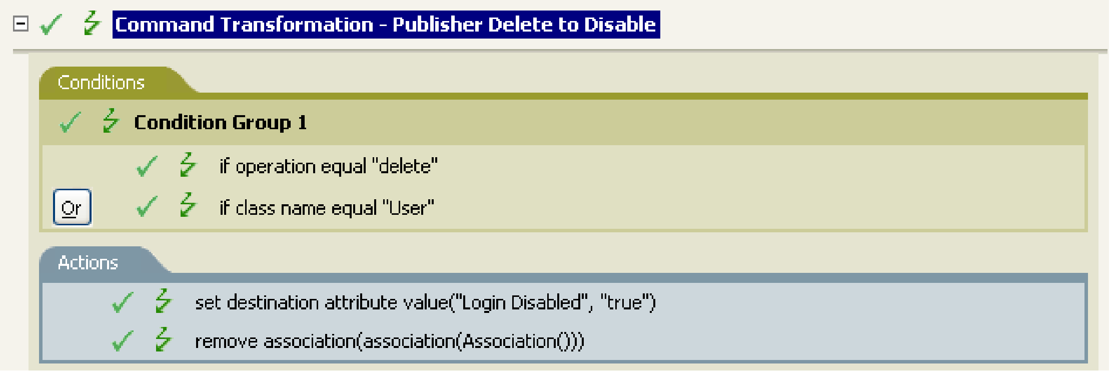
Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

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 Identity Manager. For more information, see Command Transformation - Publisher Delete to Disable (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeletetodisable.html#prdeletetodisable). To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).



Command Transformation - Publisher Delete to Disable

Conditions

Condition Group 1

if operation equal "delete"

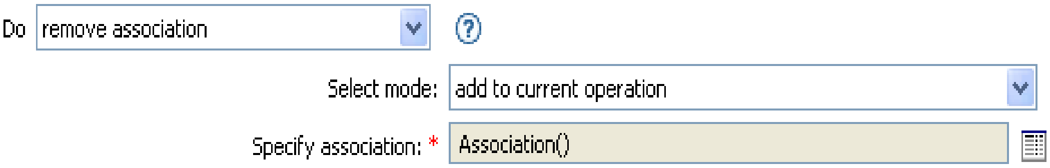
Or

if class name equal "User"

Actions

set destination attribute value("Login Disabled", "true")

remove association(association(Association()))



Do: remove association

Select mode: add to current operation

Specify association: * Association()

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 Expansion (page 290).

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 Expansion (page 290).

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Select Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

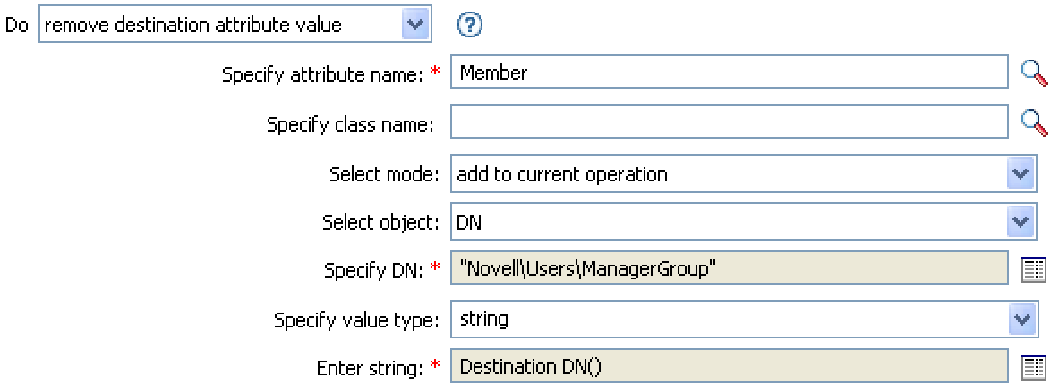
Value Type

Specify the syntax of the new attribute value.

String

Specify the value of the new attribute.

Example



Do: remove destination attribute value

Specify attribute name: * Member

Specify class name:

Select mode: add to current operation

Select object: DN

Specify DN: * "Novell\Users\ManagerGroup"

Specify value type: string

Enter string: * Destination DN()

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 Expansion (page 290).

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 Expansion (page 290).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

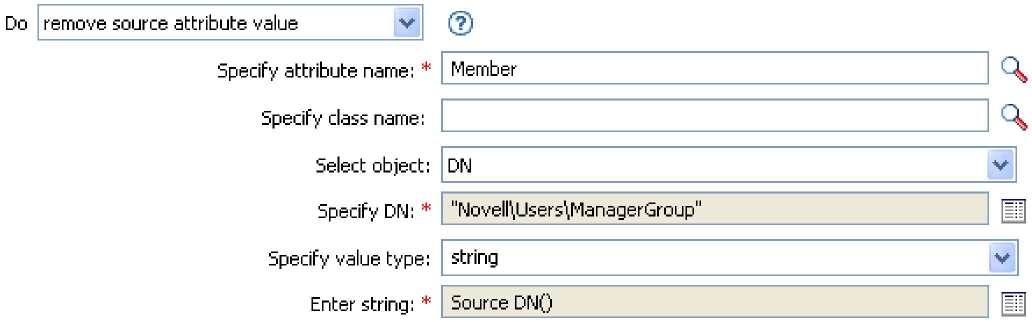
Value Type

Specify the syntax of the attribute value to be removed.

String

Specify the attribute value to be removed.

Example



Do remove source attribute value

Specify attribute name: * Member

Specify class name:

Select object: DN

Specify DN: * "Novell\Users\ManagerGroup"

Specify value type: string

Enter string: * Source DN()

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.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

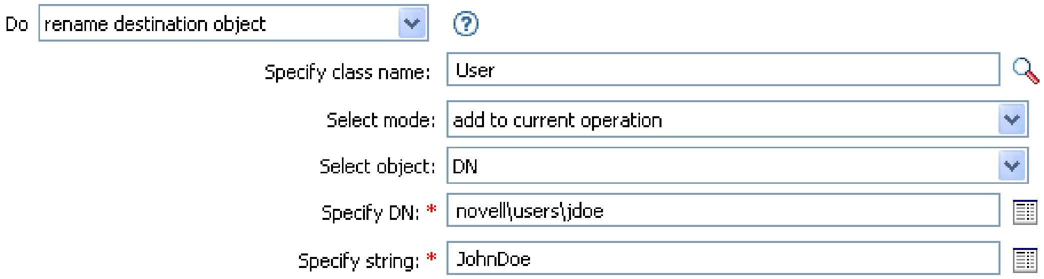
Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

String

Specify the new name of the object.

Example



Do rename destination object

Specify class name: User

Select mode: add to current operation

Select object: DN

Specify DN: * novell\users\jdoe

Specify string: * JohnDoe

Rename Operation Attribute

Renames all occurrences of an attribute within the current operation.

Fields

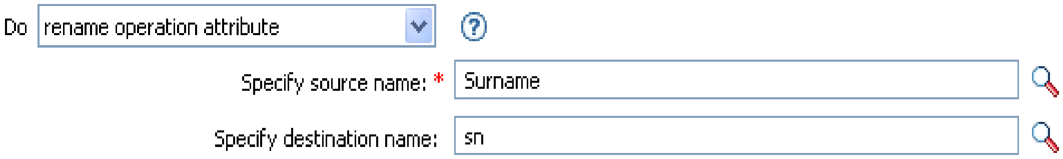
Source Name

Specify the original attribute name. Supports variable expansion. For more information, see Variable Expansion (page 290).

Destination Name

Specify the new attribute name. Supports variable expansion. For more information, see Variable Expansion (page 290).

Example



Do rename operation attribute

Specify source name: * Surname

Specify destination name: sn

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.

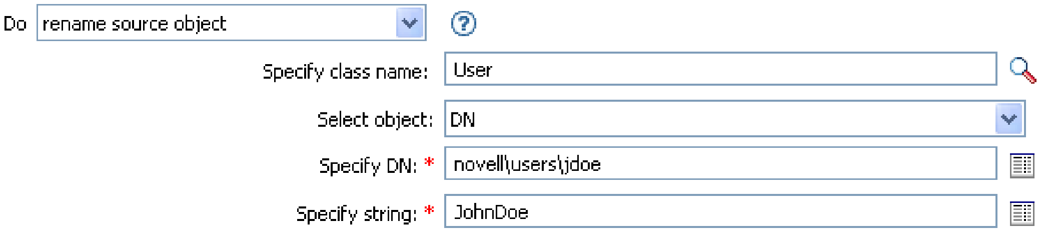
Select Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

String

Specify the new name of the object.

Example



Do: rename source object

Specify class name: User

Select object: DN

Specify DN: * novell\users\jdoe

Specify string: * JohnDoe

Send Email

Sends an e-mail notification.

Fields

ID

(Optional) Specify the User ID in the SMTP system sending the message. Supports variable expansion. For more information, see Variable Expansion (page 290).

Server

Specify the SMTP server name. Supports variable expansion. For more information, see Variable Expansion (page 290).

Message Type

Select the e-mail message type.

Password

(Optional) Specify the SMTP server account password.

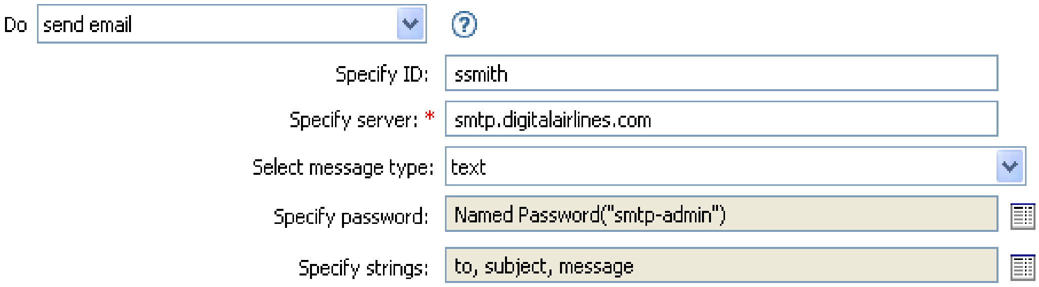
IMPORTANT: You can store the SMTP server account password as a Named Password on the driver object. This allows the password to be encrypted; otherwise you enter the password and it is stored in clear text. For more information on Named Passwords, see Using Named Password in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

Strings

Specify the values containing the various e-mail addresses, subject, and message. The following table lists valid named string arguments:

String Name	Description
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.
from	Specifies the address to be used as the originating e-mail address.
reply-to	Specifies the address to be used as the e-mail message reply address.
subject	Specifies the e-mail subject.
message	Specifies the content of the e-mail message.
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.

Example



Do send email

Specify ID: ssmith

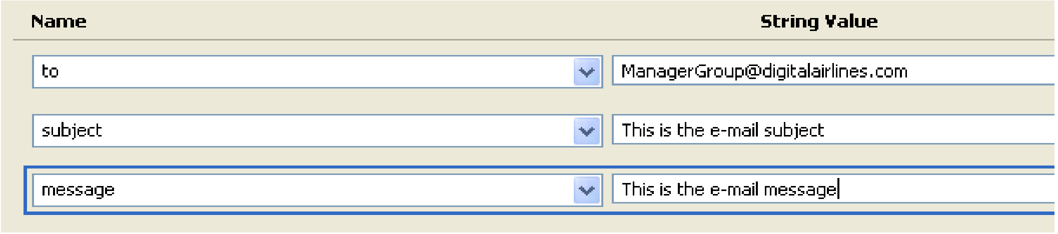
Specify server: * smtp.digitalairlines.com

Select message type: text

Specify password: Named Password("smtp-admin")

Specify strings: to, subject, message

The following is an example of the Named String Builder being used to provide the strings argument:



Name	String Value
to	ManagerGroup@digitalairlines.com
subject	This is the e-mail subject
message	This is the e-mail message

Send Email from Template

Generates an e-mail notification using a template.

Fields

Notification DN

Specify the slash form DN of the SMTP notification configuration object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Template DN

Specify the slash form DN of the e-mail template object. Supports variable expansion. For more information, see Variable Expansion (page 290).

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 Using Named Passwords in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

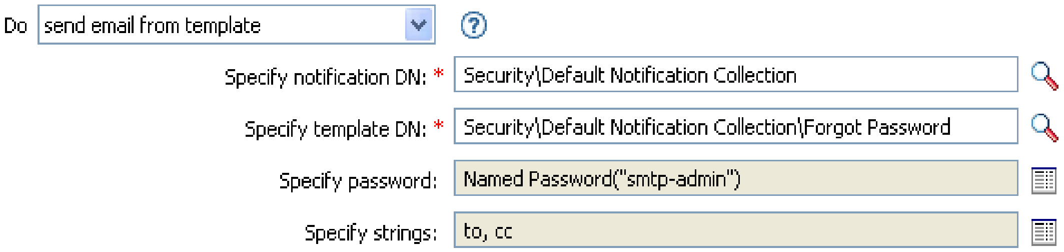
Strings

Specify additional fields for the e-mail message. The following table contains reserved field names, which specify the various e-mail addresses:

String Name	Description
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.
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.

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

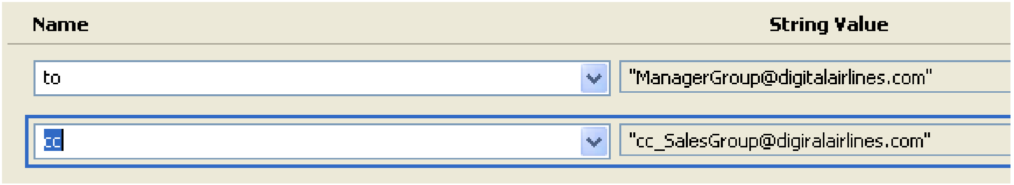
Specify notification DN: * Security\Default Notification Collection

Specify template DN: * Security\Default Notification Collection\Forgot Password

Specify password: Named Password("smtp-admin")

Specify strings: to, cc

The following is an example of the Named String Builder, used to provide the strings argument:



Name	String Value
to	"ManagerGroup@digitalairlines.com"
cc	"cc_SalesGroup@digiralairlines.com"

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 Expansion (page 290).

Write Back

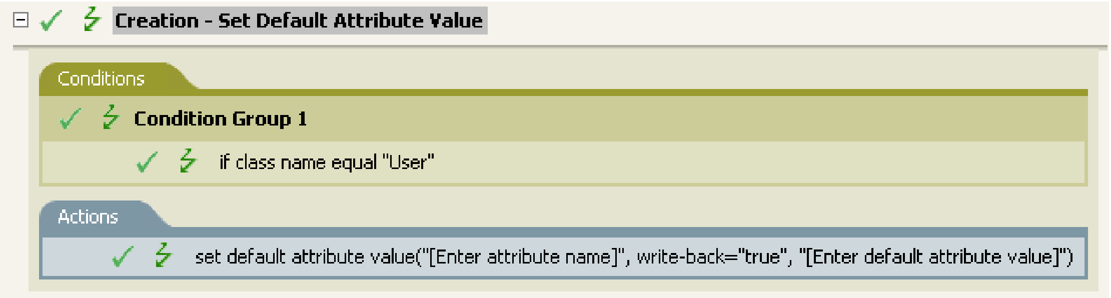
Select whether or not to also write back the default values to the source data store.

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 Identity Manager. For more information, see Creation- Set Default Attribute Value (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdefaultattr.html#prdefaultattr). To view the policy in XML, see predef_creation_set_default_attribute_value.xml (../samples/predef_creation_set_default_attribute_value.xml).



Creation - Set Default Attribute Value

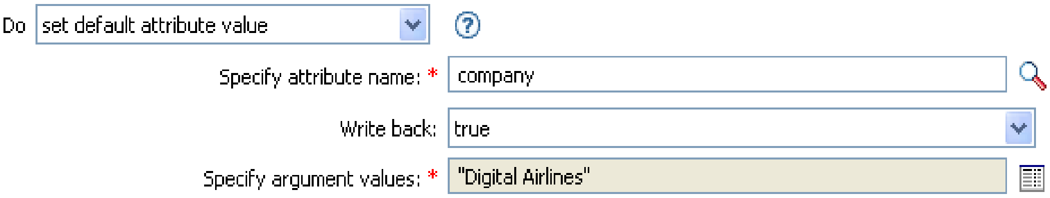
Conditions

Condition Group 1

if class name equal "User"

Actions

set default attribute value("[Enter attribute name]", write-back="true", "[Enter default attribute value]")

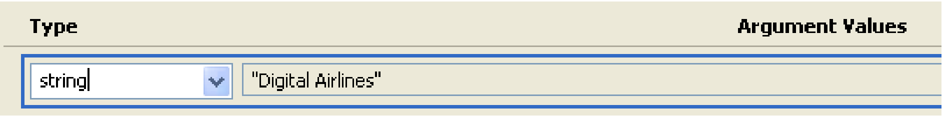


Do: set default attribute value

Specify attribute name: * company

Write back: true

Specify argument values: * "Digital Airlines"



Type	Argument Values
string	"Digital Airlines"

To build the value, the Argument Value List Builder is launched. See Argument Value List Builder (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/pbvaluebuilder.html#pbvaluebuilder) 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 Expansion (page 290).

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 Expansion (page 290).

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

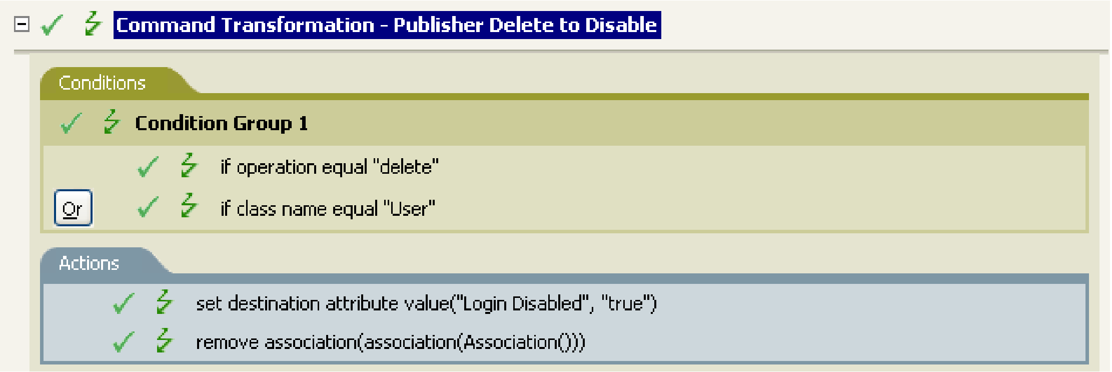
Select the syntax of the attribute value to set.

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 Identity Manager. For more information, see Command Transformation - Publisher Delete to Disable (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeletetodisable.html#prdeletetodisable). To view the policy in XML, see predef_command_delete_to_disable.xml (../samples/predef_command_delete_to_disable.xml).



Command Transformation - Publisher Delete to Disable

Conditions

Condition Group 1

if operation equal "delete"

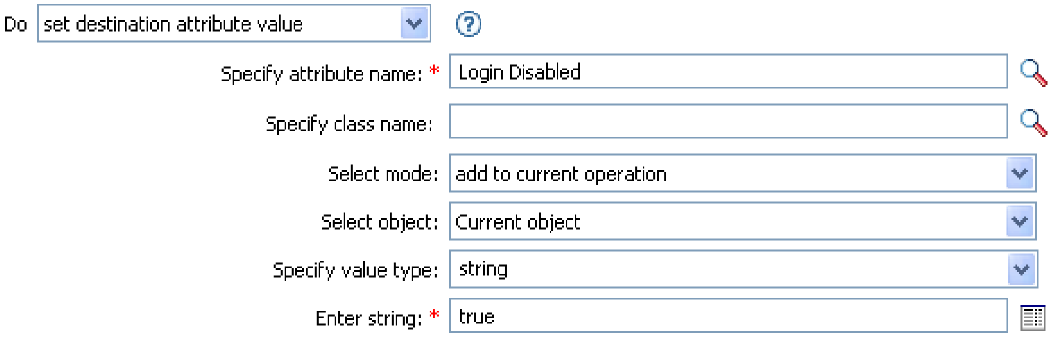
Or

if class name equal "User"

Actions

set destination attribute value("Login Disabled", "true")

remove association(association(Association()))



Do: set destination attribute value

Specify attribute name: * Login Disabled

Specify class name:

Select mode: add to current operation

Select object: Current object

Specify value type: string

Enter string: * true

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 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/pbargbuilder.html#pbargbuilder) 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.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Object

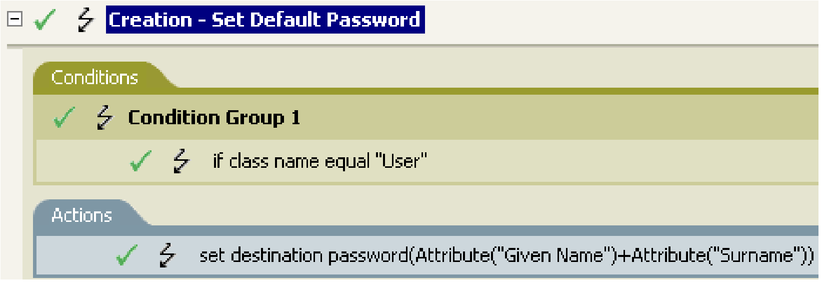
Select the target object. This object can be the current object, or can be specified by an DN or an association.

String

Specify the password to be set.

Example

The example sets a default password for the User object that is created. The rule is from the predefined rules that come with Identity Manager. For more information, see Creation- Set Default Password (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdefaultpassword.html#prdefaultpassword). To view the policy in XML, see predef_creation_set_default_password.xml (../samples/predef_creation_set_default_password.xml).



⊟ ✓ Creation - Set Default Password

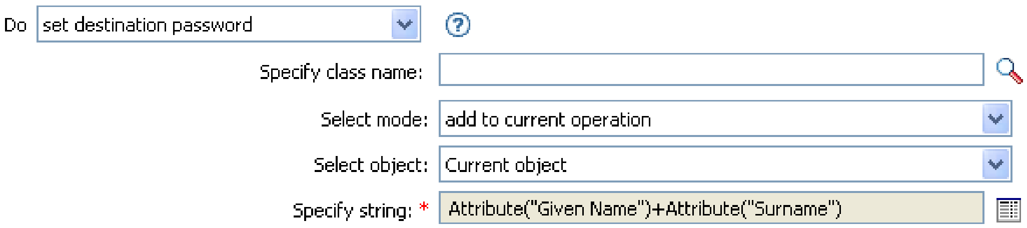
Conditions

✓ Condition Group 1

✓ if class name equal "User"

Actions

✓ set destination password(Attribute("Given Name")+Attribute("Surname"))



Do: set destination password

Specify class name:

Select mode: add to current operation

Select object: Current object

Specify string: * Attribute("Given Name")+Attribute("Surname")

When a User object is created, the password is set to the Given Name attribute plus the Surname attribute.

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 Expansion (page 290).

Variable Scope

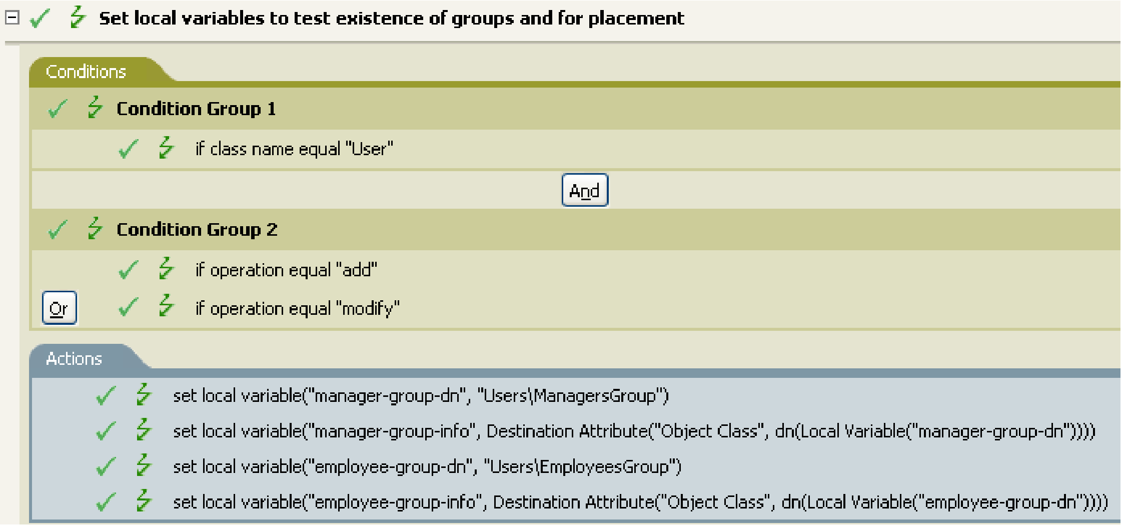
Select the scope of the local variable. This can be set to the driver or to the policy. Supports variable expansion. For more information, see Variable Expansion (page 290).

Variable Type

Select the type of local variable. This can be a string, an XPath 1.0 node set, or a Java object.

Example

The example adds a User object to the appropriate group, Employee or Manager, 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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 003-AddCreateGroups.xml (../samples/003-Command-AddCreateGroups.xml).



Set local variables to test existence of groups and for placement

Condition Group 1

if class name equal "User"

Condition Group 2

if operation equal "add"

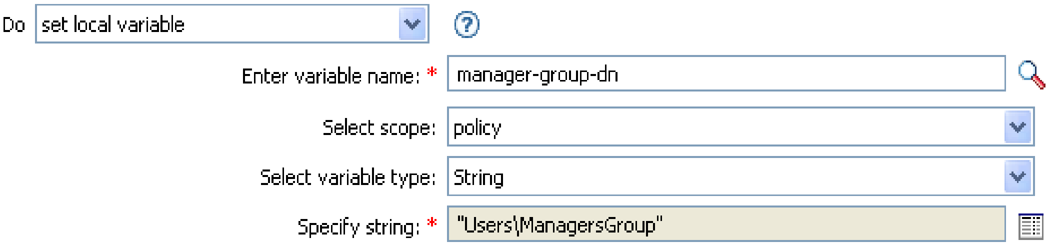
if operation equal "modify"

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"))))



Do: set local variable

Enter variable name: * manager-group-dn

Select scope: policy

Select variable type: String

Specify string: * "Users\ManagersGroup"

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 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/pbargbuilder.html#pbargbuilder) for more information.

Set Operation Association

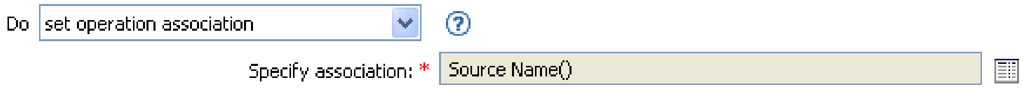
Sets the association value for the current operation.

Fields

Association

Provide the new association value.

Example



Do set operation association

Specify association: * Source Name()

Set Operation Class Name

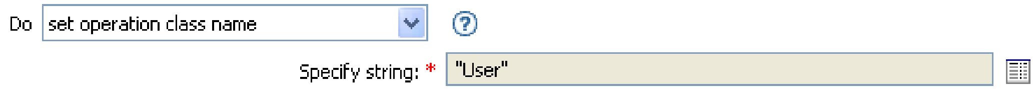
Sets the object class name for the current operation.

Fields

String

Specify the new class name.

Example



Do set operation class name

Specify string: * "User"

Set Operation Destination DN

Sets the destination DN for the current operation.

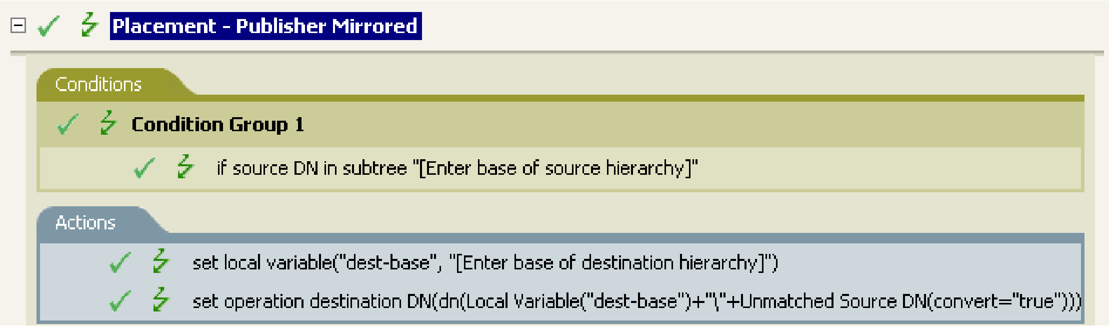
Fields

DN

Specify the new destination DN.

Example

The example places the objects in the Identity Vault 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 Identity Manager. For more information, see Placement - Publisher Mirrored (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prplacepubmirrored.html#prplacepubmirrored). To view the policy in XML, see predef_place_pub_mirrored.xml (../samples/predef_place_pub_mirrored.xml).



⊟ ✓ Placement - Publisher Mirrored

Conditions

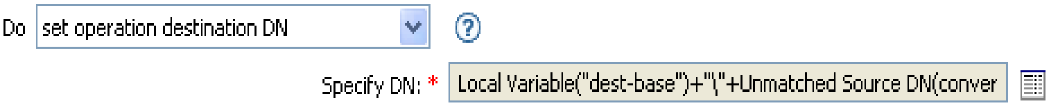
✓ Condition Group 1

✓ if source DN in subtree "[Enter base of source hierarchy]"

Actions

✓ set local variable("dest-base", "[Enter base of destination hierarchy]")

✓ set operation destination DN(dn(Local Variable("dest-base")+"\"+Unmatched Source DN(convert="true")))



Do: set operation destination DN

Specify DN: * Local Variable("dest-base")+"\"+Unmatched Source DN(conver

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

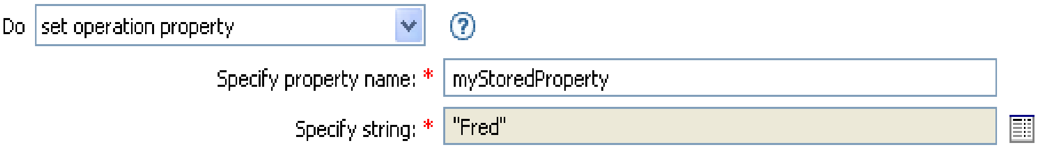
Property Name

Specify the name of the operation property. Supports variable expansion. For more information, see Variable Expansion (page 290).

String

Specify the name of the string.

Example



Do set operation property

Specify property name: * myStoredProperty

Specify string: * "Fred"

Set Operation Source DN

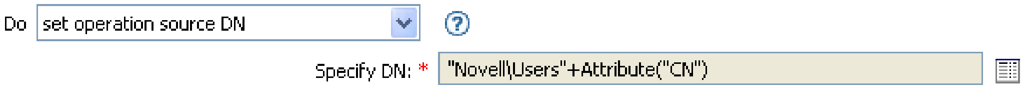
Sets the source DN for the current operation.

Fields

DN

Specify the new source DN.

Example



Do: set operation source DN

Specify DN: * "Novell\Users"+Attribute("CN")

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.

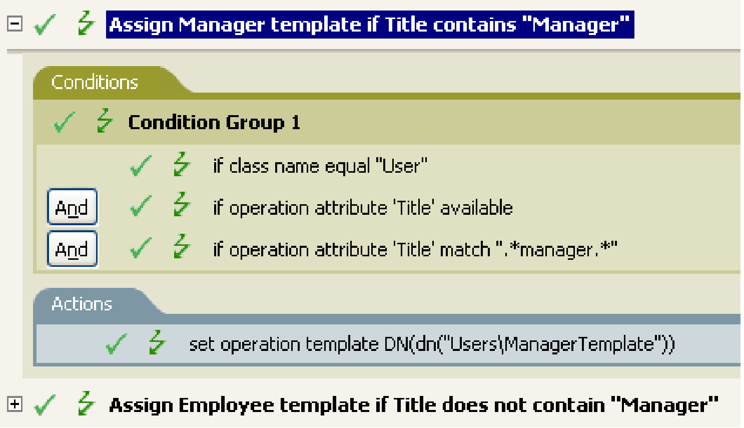
Fields

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 Tile, and it is available for download from the Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 003-Create_AssignTemplateByTitle.xml (../samples/003-Create_AssignTemplateByTitle.xml).



Assign Manager template if Title contains "Manager"

Conditions

Condition Group 1

if class name equal "User"

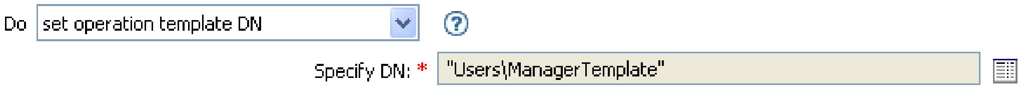
And if operation attribute 'Title' available

And if operation attribute 'Title' match ".*manager.*"

Actions

set operation template DN(dn("Users\ManagerTemplate"))

Assign Employee template if Title does not contain "Manager"



Do: set operation template DN

Specify DN: * "Users\ManagerTemplate"

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 Expansion (page 290).

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 Expansion (page 290).

Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

Value Type

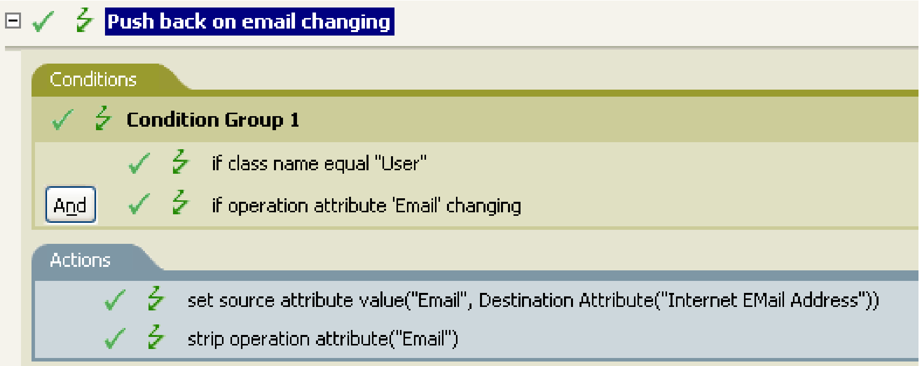
Select the syntax of the attribute value.

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Input_PushBackOnEmail (../samples/001-Input-PushBackOnEmail.xml).



Push back on email changing

Conditions

Condition Group 1

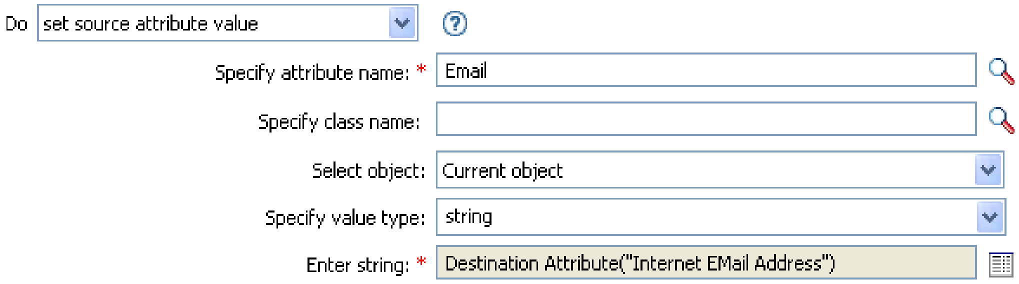
if class name equal "User"

And if operation attribute 'Email' changing

Actions

set source attribute value("Email", Destination Attribute("Internet EMail Address"))

strip operation attribute("Email")



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.

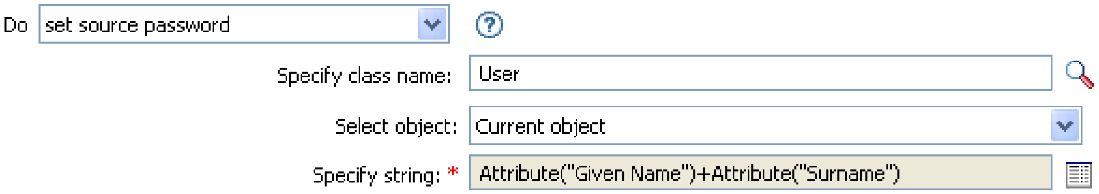
Object

Select the target object. This object can be the current object, or can be specified by an DN or an association.

String

Specify the password to be set.

Example



Do set source password

Specify class name: User

Select object: Current object

Specify string: * Attribute("Given Name")+Attribute("Surname")

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 Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Target User DN

Specify the DN of the target users.

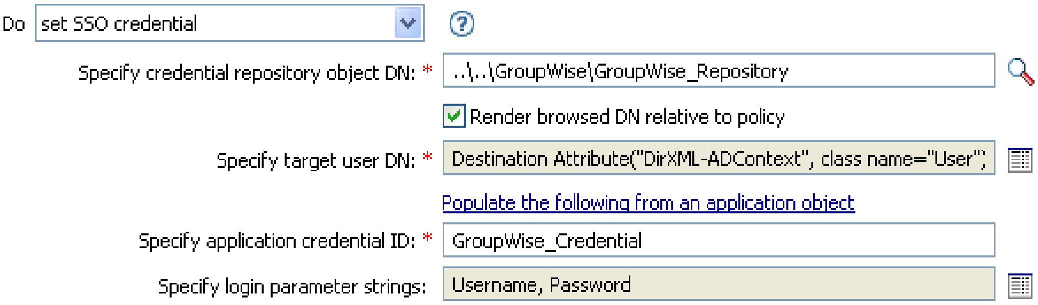
Application Credential ID

Specify the application credential that is stored in the application object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Example



Do set SSO credential

Specify credential repository object DN: * ..\..\GroupWise\GroupWise_Repository

☑ Render browsed DN relative to policy

Specify target user DN: * Destination Attribute("DirXML-ADContext", class name="User",

Populate the following from an application object

Specify application credential ID: * GroupWise_Credential

Specify login parameter strings: Username, Password

Set SSO Passphrase

Sets the Novell SecureLogin passphrase and answer when a User object is provisioned. This action is part of the Credential Provisioning policies. For more information, see Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object. Supports variable expansion. For more information, see Variable Expansion (page 290).

Target User DN

Specify the DN of the target users.

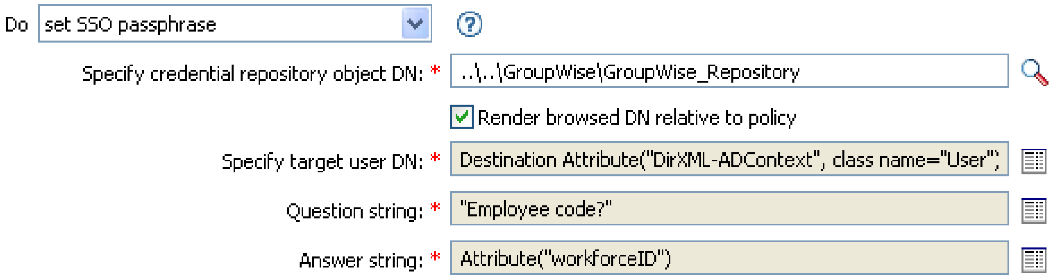
Question Strings

Specify the SecureLogin passphrase question.

Answer String

Specify the SecureLogin passphrase answer.

Example



Do: set SSO passphrase

Specify credential repository object DN: * ..\..\GroupWise\GroupWise_Repository

☑ Render browsed DN relative to policy

Specify target user DN: * Destination Attribute("DirXML-ADContext", class name="User");

Question string: * "Employee code?"

Answer string: * Attribute("workforceID")

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 Expansion (page 290).

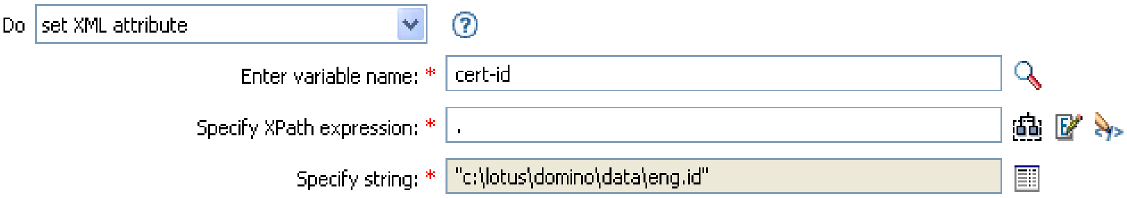
XPath Expression

XPath 1.0 expression that returns a node set containing the elements on which the XML attribute should be set.

String

Specify the value of the XML attribute.

Example



Do set XML attribute

Enter variable name: * cert-id

Specify XPath expression: * .

Specify string: * "c:\lotus\domino\data\eng.id"

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 Expansion (page 290).

Message

Provide the status message using the Argument Builder.

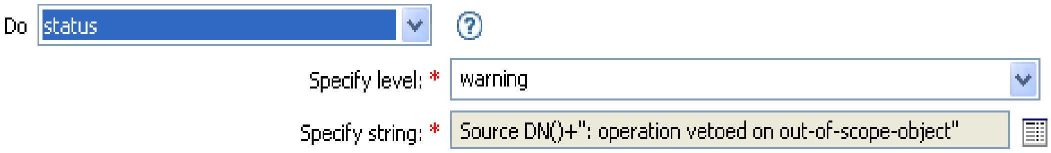
Remarks

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



Do status

Specify level: * warning

Specify string: * Source DN()+": operation vetoed on out-of-scope-object"

Start Workflow

Starts the workflow specified by workflow-id for the recipient DN on the User Application server specified by a URL and 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. The number of the strings and the names used are dependent on the workflow to be started.

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.

If any type of 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 Expansion (page 290).

User Application URL

Specify the URL of the User Application server where the workflow will run. Supports variable expansion. For more information, see Variable Expansion (page 290).

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 Expansion (page 290).

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

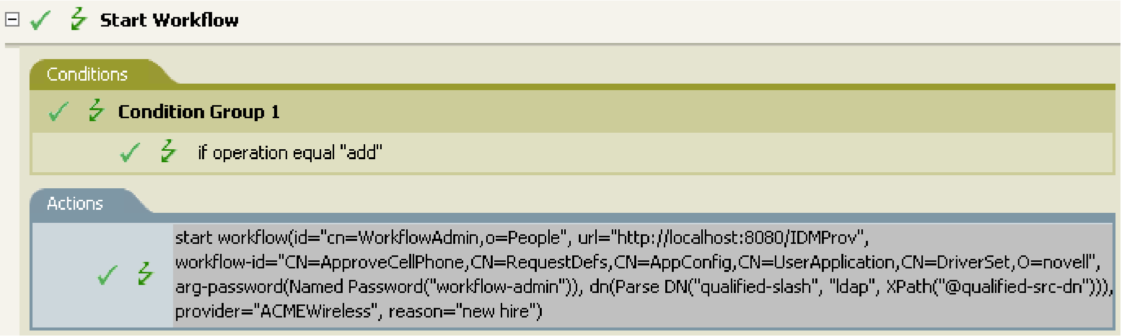
Specify the DN of the recipient of the workflow in LDAP format.

Additional Arguments

Specify the arguments for the workflow. The arguments are defined on the workflow. Some of the arguments might be required for the workflow to start. It depends upon how the workflow is defined.

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).



⊟ ✓ Start Workflow

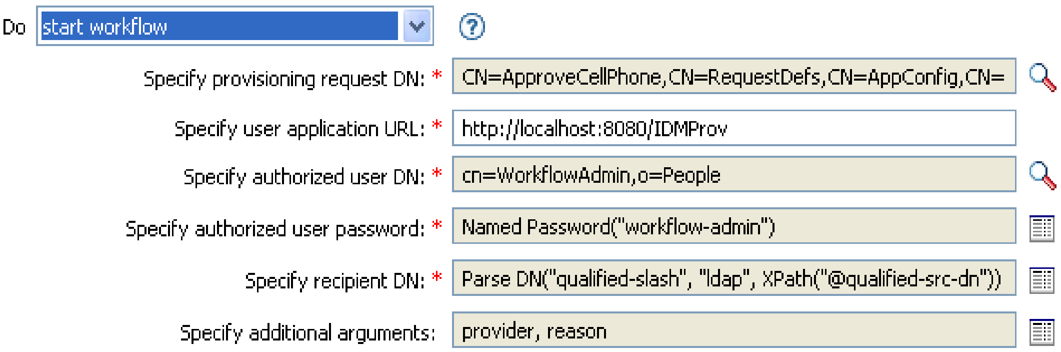
Conditions

✓ Condition Group 1

✓ if operation equal "add"

Actions

✓ start workflow(id="cn=WorkflowAdmin,o=People", url="http://localhost:8080/IDMProv", workflow-id="CN=ApproveCellPhone,CN=RequestDefs,CN=AppConfig,CN=UserApplication,CN=DriverSet,O=novell", arg-password(Named Password("workflow-admin")), dn(Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))), provider="ACMEWireless", reason="new hire")



Do: start workflow

Specify provisioning request DN: *	CN=ApproveCellPhone,CN=RequestDefs,CN=AppConfig,CN=
Specify user application URL: *	http://localhost:8080/IDMProv
Specify authorized user DN: *	cn=WorkflowAdmin,o=People
Specify authorized user password: *	Named Password("workflow-admin")
Specify recipient DN: *	Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))
Specify additional arguments:	provider, reason

Strip Operation Attribute

Strips all occurrences of an attribute from the current operation.

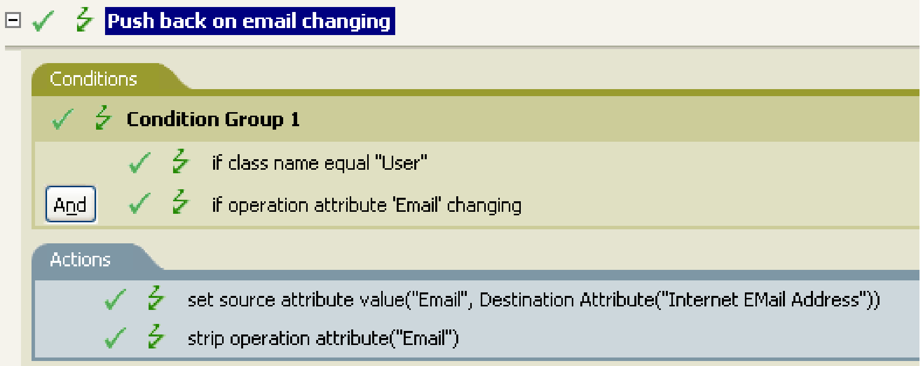
Fields

Name

Specify the name of the attribute to be stripped. Supports variable expansion. For more information, see Variable Expansion (page 290).

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Input-PushBackOnEmail.xml (../samples/001-Input-PushBackOnEmail.xml).



Push back on email changing

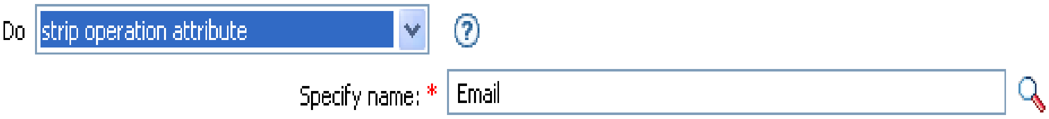
Condition Group 1

if class name equal "User"

And if operation attribute 'Email' changing

set source attribute value("Email", Destination Attribute("Internet EMail Address"))

strip operation attribute("Email")



Do: strip operation attribute

Specify name: * Email

The action strips the attribute of Email. The value that is kept is what was in the destination Email attribute.

Strip XPath

Strips nodes selected by an XPath 1.0 expression.

Fields

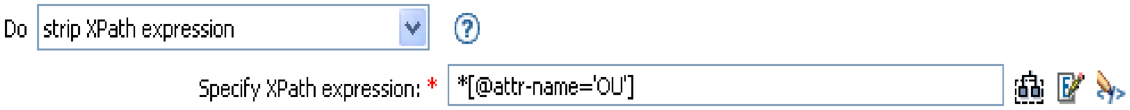
XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the nodes to be stripped.

Remarks

For more information on using XPath expression with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example



Do strip XPath expression

Specify XPath expression: * *[@attr-name='OU']

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 Versioning Information in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

Color

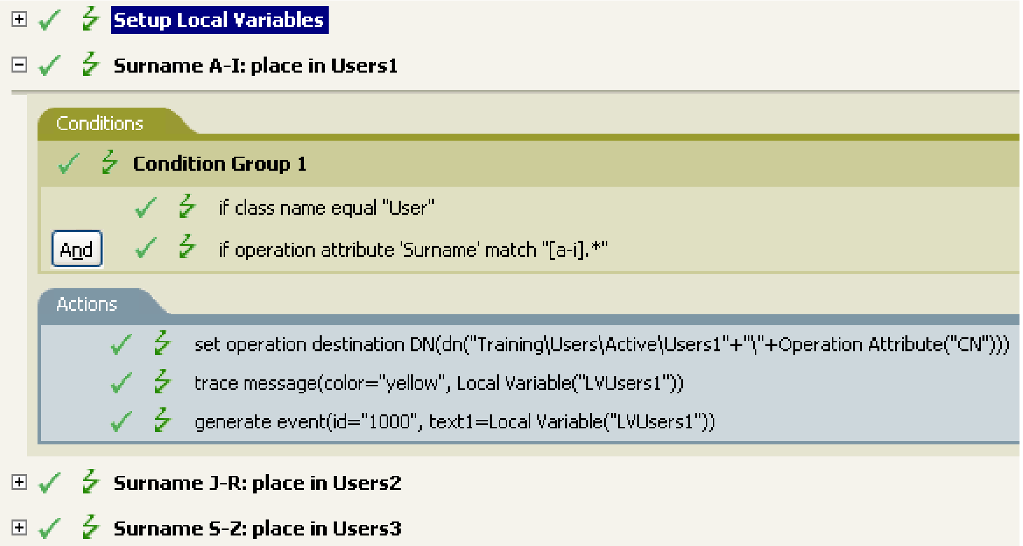
Select the color of the trace message.

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 Novell Audit or Sentinel event. The Trace Message action is used to send a trace message into DSTRACE. The policy name is Policy to Place by Surname and it is available for download from the Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). 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

Condition Group 1

if class name equal "User"

if operation attribute 'Surname' match "[a-i].*"

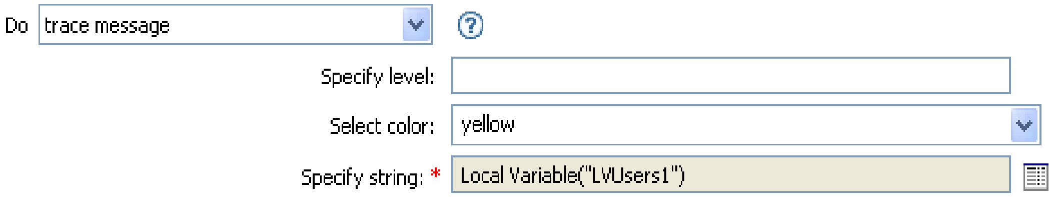
set operation destination DN(dn("Training\Users\Active\Users1"+"\"+Operation Attribute("CN")))

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 trace message

Specify level:

Select color: yellow

Specify string: * Local Variable("LVUsers1")

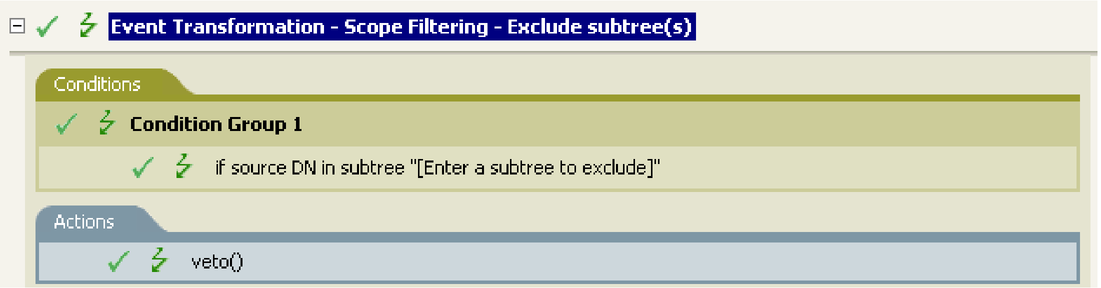
The action sends a trace message to DSTRACE. The contents of the local variable is LVUsers1 and it shows up in yellow in DSTRACE.

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 Identity Manager. For more information, see Event Transformation - Scope Filtering - Exclude Subtrees (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prfilterexcludesubtree.html#prfilterexcludesubtree) from the predefined rules. To view the policy in XML, see predef_transformation_filter_exclude_subtress.xml (../samples/predef_transformation_filter_exclude_subtrees.xml).



Event Transformation - Scope Filtering - Exclude subtree(s)

Conditions

Condition Group 1

if source DN in subtree "[Enter a subtree to exclude]"

Actions

veto()

Do veto

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.

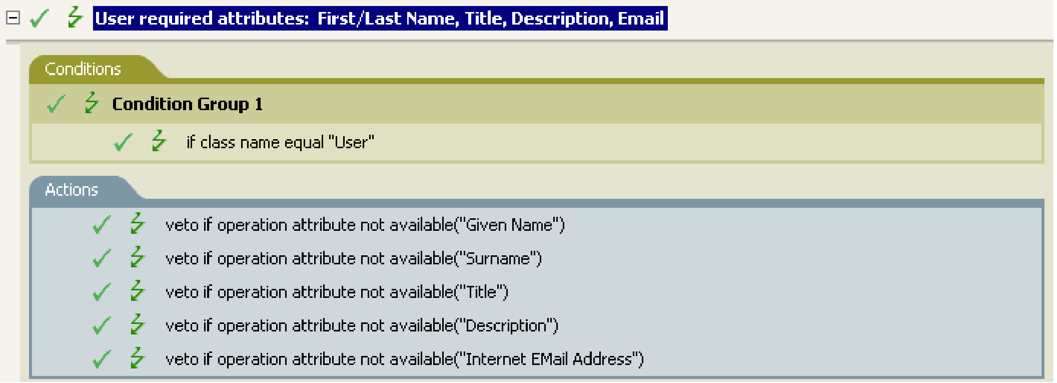
Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 290).

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Create-RequiredAttrs.xml (../samples/001-Create-RequiredAttrs.xml).



User required attributes: First/Last Name, Title, Description, Email

Condition Group 1

if class name equal "User"

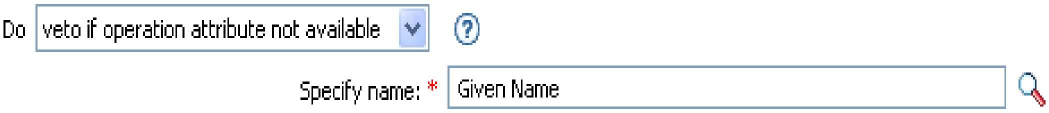
veto if operation attribute not available("Given Name")

veto if operation attribute not available("Surname")

veto if operation attribute not available("Title")

veto if operation attribute not available("Description")

veto if operation attribute not available("Internet EMail Address")



Do veto if operation attribute not available

Specify name: * Given Name

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

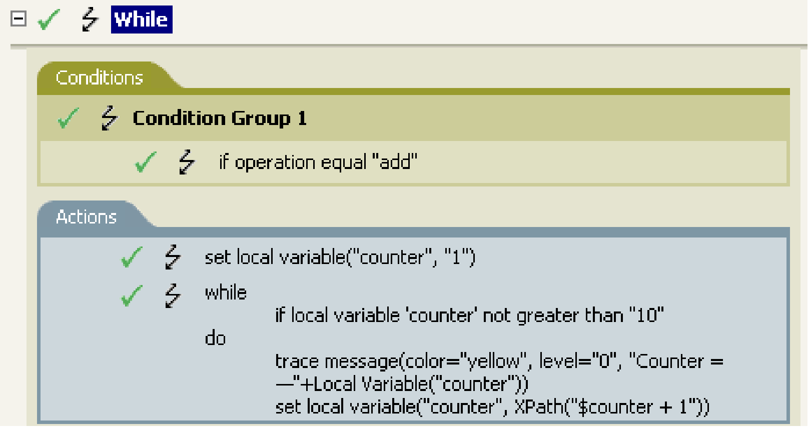
Conditions

Specify the condition to be evaluated.

Actions

Specify the actions to be repeated if the conditions evaluate to True.

Example



While

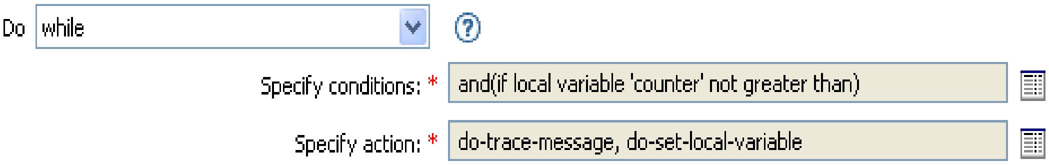
Conditions

Condition Group 1

if operation equal "add"

Actions

```
set local variable("counter", "1")
while
          if local variable 'counter' not greater than "10"
do
          trace message(color="yellow", level="0", "Counter =
          —"+Local Variable("counter"))
          set local variable("counter", XPath("$counter + 1"))
```



Do: while

Specify conditions: * and(if local variable 'counter' not greater than)

Specify action: * do-trace-message, do-set-local-variable

Variable Expansion

Allows for the use of dynamic variables in the action

Remark

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 or global variable with the given name. $<variable-name>$ must be legal variable name. For information on what is 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, it should be escaped with an additional $ (for example, You owe me $$100.00).

Noun Tokens

14

Noun tokens expand to values that are derived from the current operation, the source or destination data stores, or some external source.

This section contains detailed information about all noun tokens that are available through using the Policy Builder interface.

Added Entitlement

Expands to the values of an entitlement granted in the current operation.

Fields

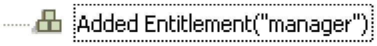
Name

Name of the entitlement. Supports variable expansion. For more information, see Variable Expansion (page 330).

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



Added Entitlement("manager")

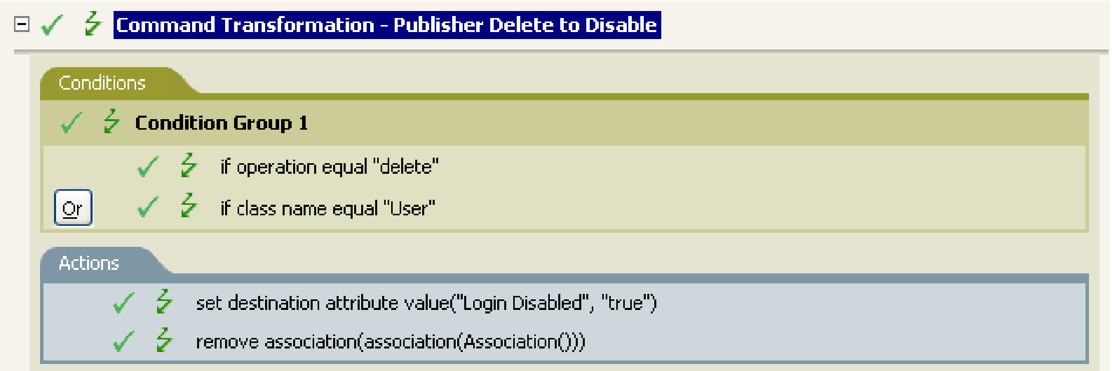
Association

Expands to the association value from the current operation.

Example

The example is from the predefined rules that come with Identity Manager. For more information on the predefined rule, see Command Transformation - Publisher Delete to Disable (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeletetodisable.html#prdeletetodisable).

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_diable.xml).



Command Transformation - Publisher Delete to Disable

Conditions

Condition Group 1

if operation equal "delete"

Or if class name equal "User"

Actions

set destination attribute value("Login Disabled", "true")

remove association(association(Association()))

Association()

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 Expansion (page 330).

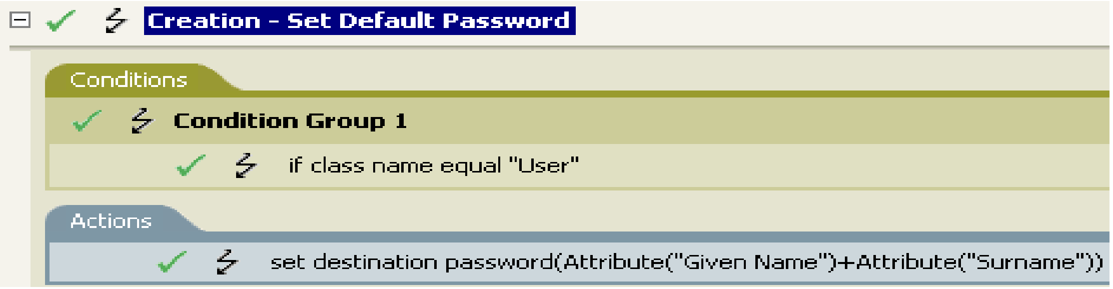
Remarks

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 Identity Manager. For more information, see Creation - Set Default Password (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdefaultpassword.html#prdefaultpassword).

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).



Creation - Set Default Password

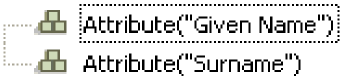
Conditions

Condition Group 1

if class name equal "User"

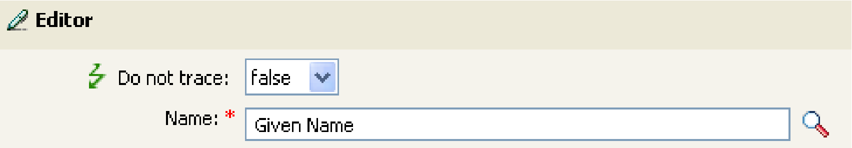
Actions

set destination password(Attribute("Given Name")+Attribute("Surname"))



Attribute("Given Name")

Attribute("Surname")



Editor

Do not trace: false

Name: * Given Name

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/).

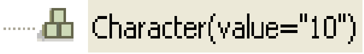
Fields

Character Value

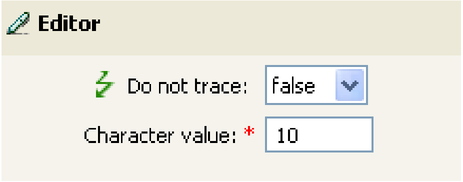
The Unicode code point of the character. Supports variable expansion. For more information, see Variable Expansion (page 330).

A hexadecimal number can be specified if it is prefixed with `0x`, as in C-based programming languages.

Example



Character(value="10")



Editor

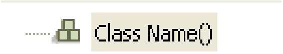
Do not trace: false

Character value: * 10

Class Name

Expands to the object class name from the current operation.

Example



Class Name()

Destination Attribute

Expands to the specified attribute value an object.

Fields

Name

Name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 330).

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 Expansion (page 330).

Select Object

Select Current Object, DN, or Association.

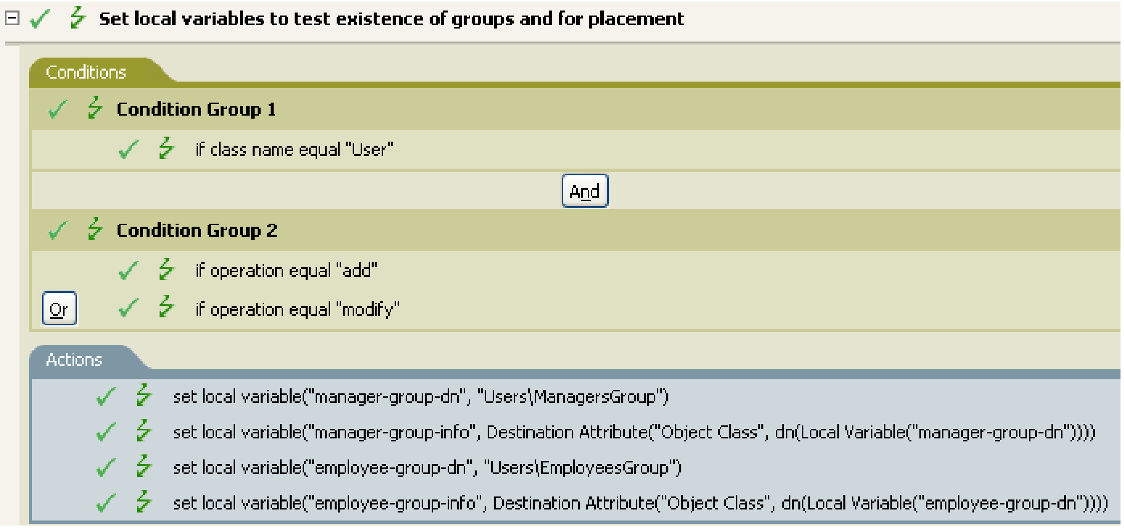
Remarks

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

The example is from the Govern Groups for User Based on Title policy, which is available for download from the Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). 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.



Set local variables to test existence of groups and for placement

Conditions

Condition Group 1

if class name equal "User"

And

Condition Group 2

if operation equal "add"

Or if operation equal "modify"

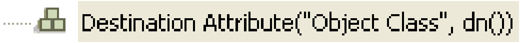
Actions

set local variable("manager-group-dn", "Users\ManagersGroup")

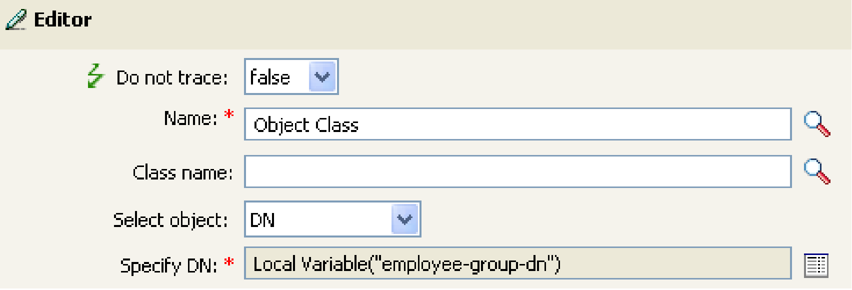
set local variable("manager-group-info", Destination Attribute("Object Class", dn(Local Variable("manager-group-dn"))))

set local variable("employee-group-dn", "Users\EmployeesGroup")

set local variable("employee-group-info", Destination Attribute("Object Class", dn(Local Variable("employee-group-dn"))))



Destination Attribute("Object Class", dn())



Editor

Do not trace:	false
Name: *	Object Class
Class name:	
Select object:	DN
Specify DN: *	Local Variable("employee-group-dn")

You build the Destination Attribute through the Editor. In this example, the attribute of Object Class is set. 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

Convert

Select whether or not to convert the DN to the format used by the source data store.

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

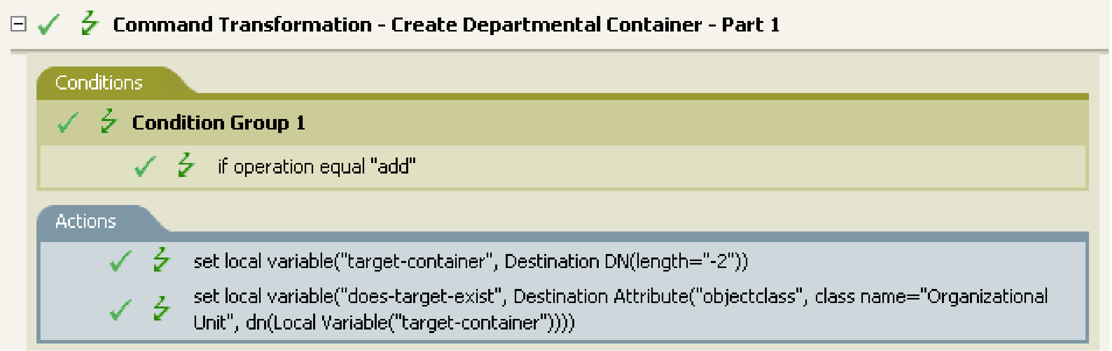
Specify the number of RDN segments to include. Negative numbers are interpreted as (total # of segments + length) + 1. For example, for a DN with 5 segments a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

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 Identity Manager. For more information, see Command Transformation - Create Departmental Container - Part 1 and Part 2 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeptcontainer.html#prdeptcontainer). To view the policy in XML, see predef_command_create_dept_container1.xml (../samples/predef_command_create_dept_container1.xml).



⊟ ✓ Command Transformation - Create Departmental Container - Part 1

Conditions

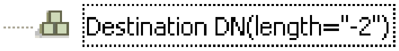
✓ Condition Group 1

✓ if operation equal "add"

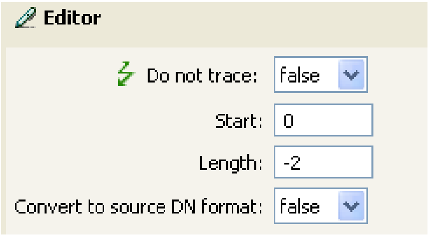
Actions

✓ 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"))))



Destination DN(length="-2")



Do not trace: false

Start: 0

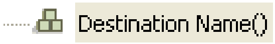
Length: -2

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



Destination Name()

Document

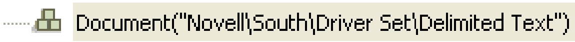
Reads the XML document pointed to by the URI and returns the document node in a node set. The URI can be relative to the URI of the including policy. With any error, the result is an empty node set.

Fields

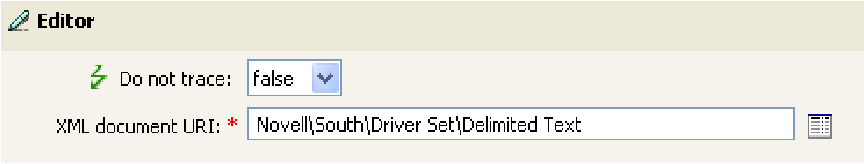
XML Document URI

Specify the XML document URI.

Example



Document("Novell\South\Driver Set\Delimited Text")



Do not trace: false

XML document URI: * Novell\South\Driver Set\Delimited Text

Entitlement

Expands to the values of a granted entitlement from the current object.

Fields

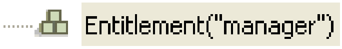
Name

Name of the entitlement. Supports variable expansion. For more information, see Variable Expansion (page 330).

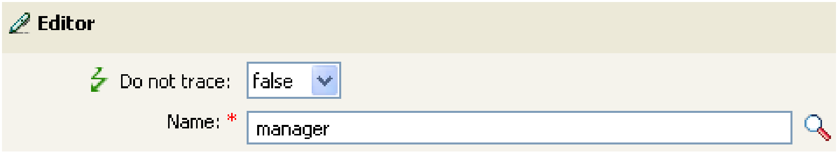
Remarks

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



Entitlement("manager")



Do not trace: false

Name: * manager

Generate Password

Generates a random password that conforms to the specified password policy.

Fields

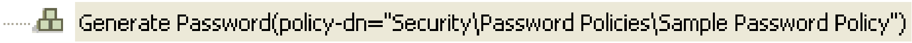
Password Policy

The DN of the password policy that receives the randomly generated password. Supports variable expansion. For more information, see Variable Expansion (page 330).

Render browsed DN relative to policy

Select whether the DN of the password policy is relative to the policy being created.

Example



```
Generate Password(policy-dn="Security\Password Policies\Sample Password Policy")
```

Global Configuration Value

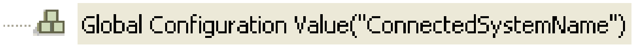
Expands to the value of a global configuration variable.

Fields

Name

Name of the global configuration value. Supports variable expansion. For more information, see Variable Expansion (page 330).

Example



Global Configuration Value("ConnectedSystemName")

Local Variable

Expands to the value of a local variable.

Fields

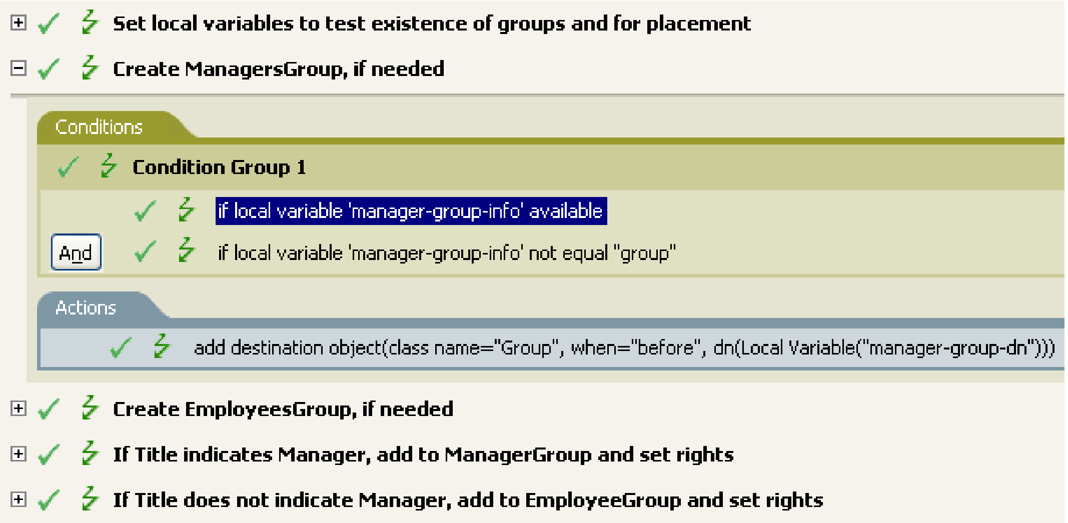
Name

Specify the name of the local variable. Supports variable expansion. For more information, see Variable Expansion (page 330).

Example

The example is from the Govern Groups for User Based on Title policy, which is available for download from the Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). 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.



⊞ ✓ Set local variables to test existence of groups and for placement

⊟ ✓ Create ManagersGroup, if needed

Conditions

✓ Condition Group 1

✓ if local variable 'manager-group-info' available

And ✓ if local variable 'manager-group-info' not equal "group"

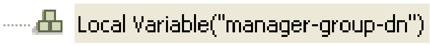
Actions

✓ add destination object(class name="Group", when="before", dn(Local Variable("manager-group-dn")))

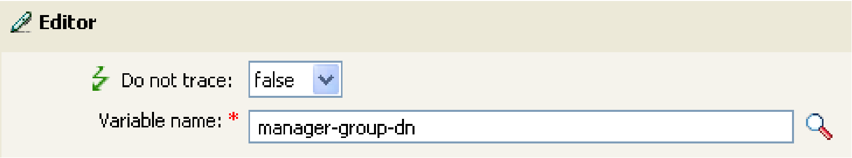
⊞ ✓ Create EmployeesGroup, if needed

⊞ ✓ If Title indicates Manager, add to ManagerGroup and set rights

⊞ ✓ If Title does not indicate Manager, add to EmployeeGroup and set rights



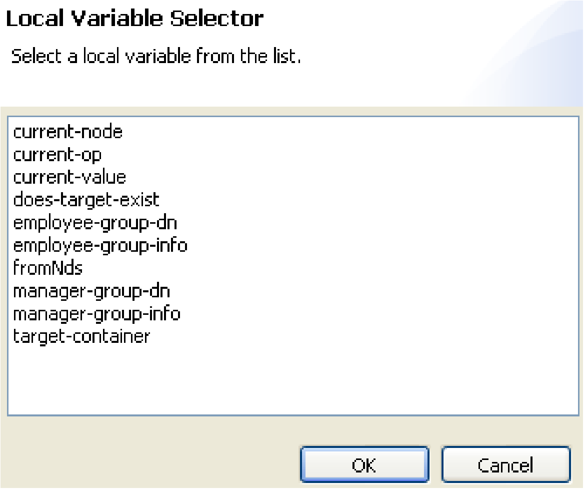
Local Variable("manager-group-dn")



Editor

Do not trace: false

Variable name: * manager-group-dn


Local Variable Selector

Select a local variable from the list.

current-node
current-op
current-value
does-target-exist
employee-group-dn
employee-group-info
fromNds
manager-group-dn
manager-group-info
target-container

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 defined group-manager-dn as DN of the manager’s group Users\ManagersGroup.

Named Password

Expands to the Named Password from the driver.

Fields

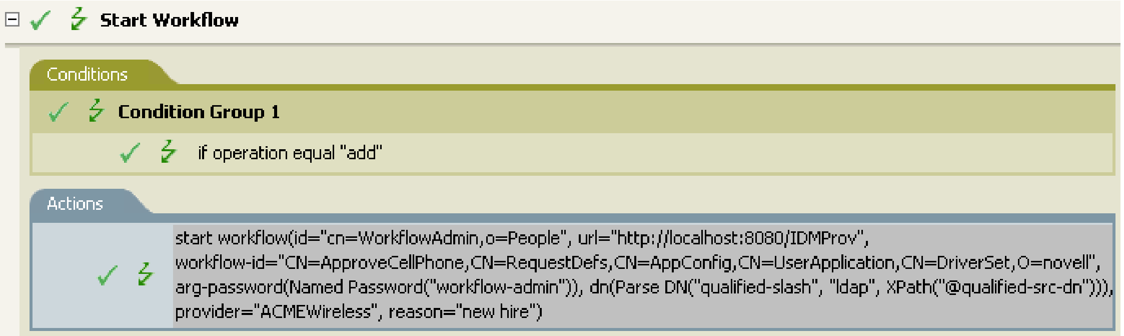
Name

Specify the Named Password. Supports variable expansion. For more information, see Variable Expansion (page 330).

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. Sometimes it is required to provide a password to allow an action to function. If you enter the password as clear text, it is a security risk. For more information on Named Passwords, see the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

The example uses the Start Workflow (page 281) 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).



Start Workflow

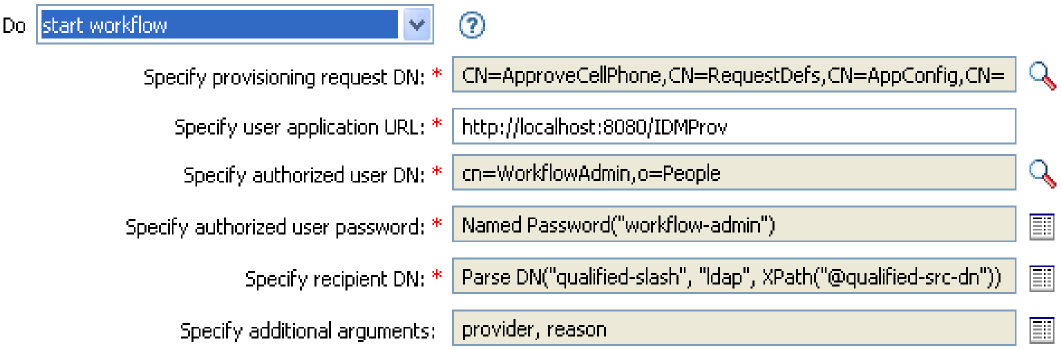
Conditions

Condition Group 1

if operation equal "add"

Actions

start workflow(id="cn=WorkflowAdmin,o=People", url="http://localhost:8080/IDMProv", workflow-id="CN=ApproveCellPhone,CN=RequestDefs,CN=AppConfig,CN=UserApplication,CN=DriverSet,O=novell", arg-password(Named Password("workflow-admin")), dn(Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))), provider="ACMEWireless", reason="new hire")



Do: start workflow

Specify provisioning request DN: * CN=ApproveCellPhone,CN=RequestDefs,CN=AppConfig,CN=

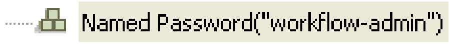
Specify user application URL: * http://localhost:8080/IDMProv

Specify authorized user DN: * cn=WorkflowAdmin,o=People

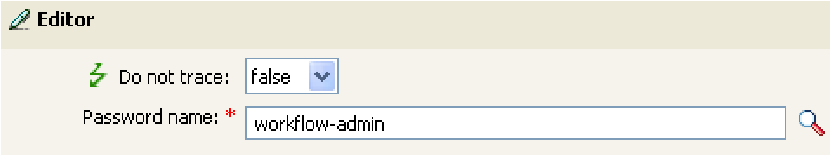
Specify authorized user password: * Named Password("workflow-admin")

Specify recipient DN: * Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))

Specify additional arguments: provider, reason



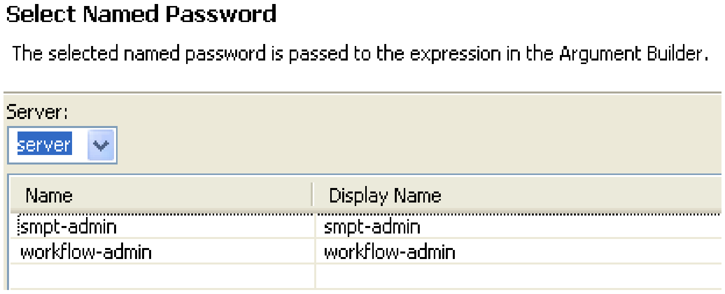
Named Password("workflow-admin")



Editor

Do not trace: false

Password name: * workflow-admin



Select Named Password

The selected named password is passed to the expression in the Argument Builder.

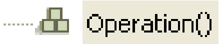
Server: server

Name	Display Name
smpt-admin	smpt-admin
workflow-admin	workflow-admin

Operation

Expands to the name of the current operation.

Example



Operation()

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.

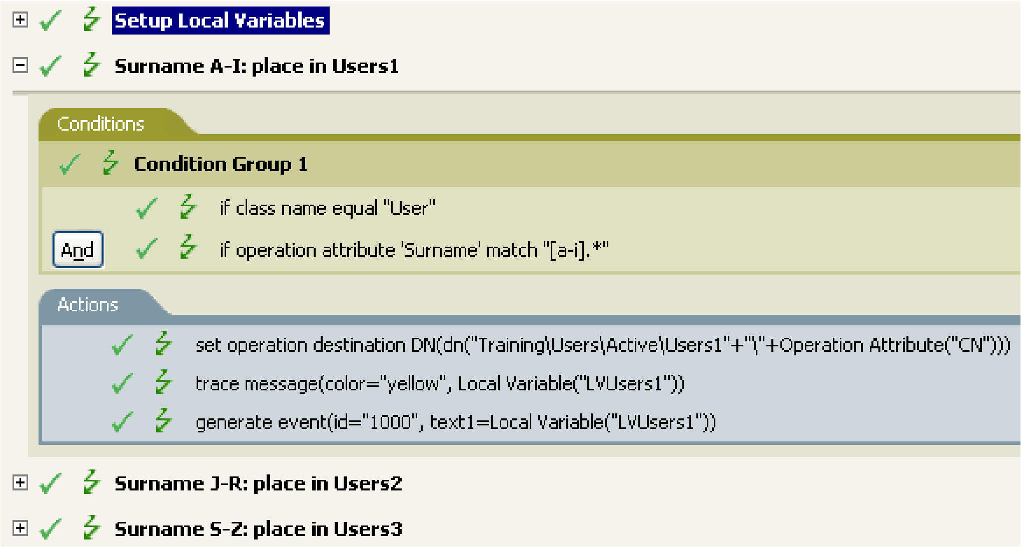
Fields

Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 330).

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 Novell Audit or Sentinel event. The policy name is Policy to Place by Surname, and it is available for download from the Novell Support Web site. For more information Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). 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

Conditions

✓ Condition Group 1

✓ if class name equal "User"

And ✓ if operation attribute 'Surname' match "[a-i].*"

Actions

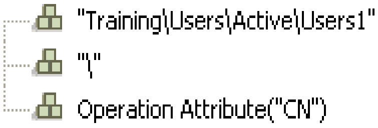
✓ set operation destination DN(dn("Training\Users\Active\Users1"+"\"+Operation Attribute("CN")))

✓ trace message(color="yellow", Local Variable("LVUsers1"))

✓ generate event(id="1000", text1=Local Variable("LVUsers1"))

⊞ ✓ Surname J-R: place in Users2

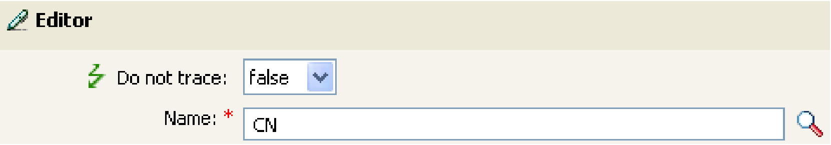
⊞ ✓ Surname S-Z: place in Users3



"Training\Users\Active\Users1"

"\"

Operation Attribute("CN")



Editor

Do not trace: false

Name: * CN

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

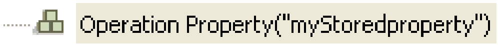
Expands to the value of the specified operation property on the current operation.

Fields

Name

Specify the name of the operation property. Supports variable expansion. For more information, see Variable Expansion (page 330).

Example

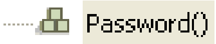


Operation Property("myStoredproperty")

Password

Expands to the password specified in the current operation.

Example



Password()

Query

Causes a query to be performed in 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 Expansion (page 330).

Select Object

Specify the base of the query. It can be the current object, DN, or an association.

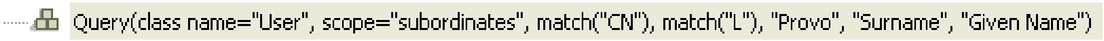
Match Attributes

Select the attributes to search for.

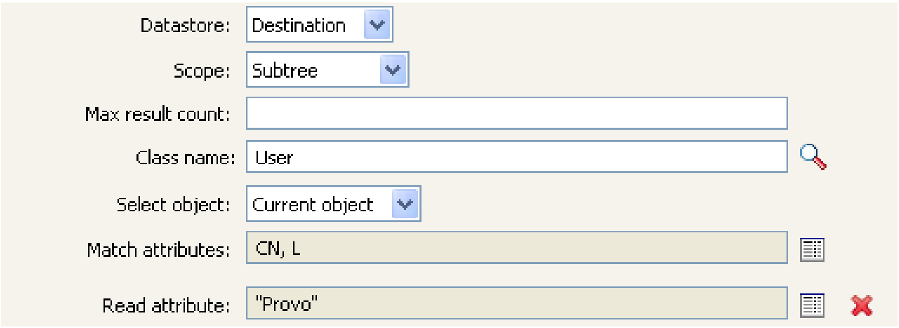
Strings

Specify the set of attributes to return. If nothing is specified, no attributes are read. Use an asterisk to read all attributes.

Example



```
Query(class name="User", scope="subordinates", match("CN"), match("L"), "Provo", "Surname", "Given Name")
```



Datastore:	Destination
Scope:	Subtree
Max result count:	
Class name:	User
Select object:	Current object
Match attributes:	CN, L
Read attribute:	"Provo"

Removed Attribute

Expands to the specified attribute value being removed in the current operation. It applies only to a modify operation.

Fields

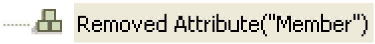
Name

Specify the name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 330).

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 Attribute("Member")

Removed Entitlements

Expands to the values of the an entitlement revoked in the current operation.

Fields

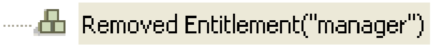
Name

Specify the name of the entitlement. Supports variable expansion. For more information, see Variable Expansion (page 330).

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



Removed Entitlement("manager")

Resolve

Resolves the DN to an association key, or the association key to a DN in the specified data store.

Fields

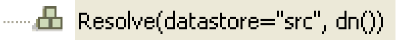
Datastore

Select the destination or source data store to be queried.

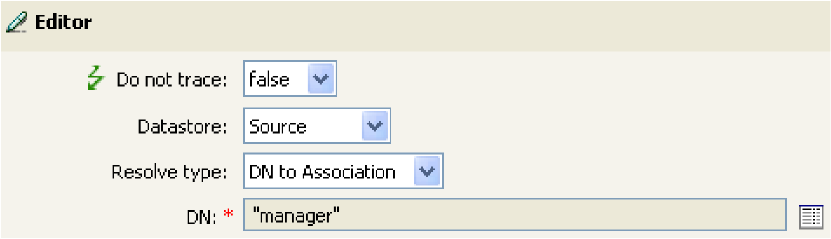
Selected Resolve Type

Select to resolve the association key to a DN or to resolve the DN to an association key.

Example



Resolve(datastore="src", dn())



Editor

Do not trace: false

Datastore: Source

Resolve type: DN to Association

DN: * "manager"

Source Attribute

Expands to the values of an attribute from an object in the source data store.

Fields

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 Expansion (page 330).

Name

Name of the attribute. Supports variable expansion. For more information, see Variable Expansion (page 330).

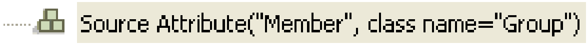
Object

Select the source object. This object can be the current object, or can be specified by a DN or an association.

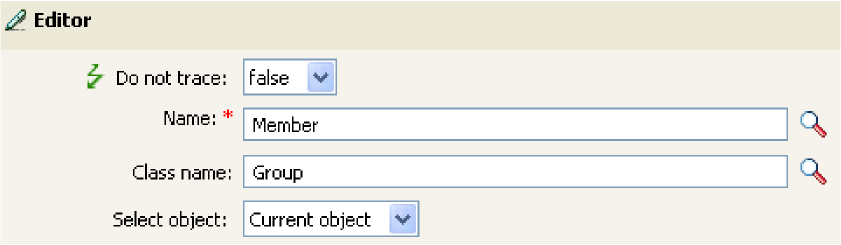
Remarks

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 Attribute("Member", class name="Group")



Editor

Do not trace: false

Name: * Member

Class name: Group

Select object: Current object

Source DN

Expands to the source DN from the current operation.

Fields

Convert

Select whether or not to convert the DN to the format used by the destination data store.

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

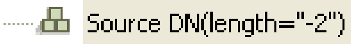
Length

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.

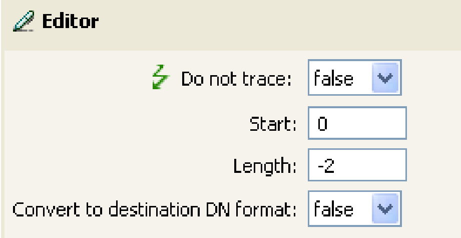
Remarks

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 DN(length="-2")



Editor

Do not trace: false

Start: 0

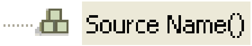
Length: -2

Convert to destination DN format: false

Source Name

Expands to the unqualified relative distinguished name (RDN) of the source DN specified in the current operation.

Example



Source Name()

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 Expansion (page 330).

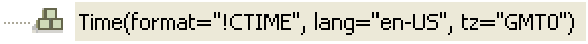
Language

Specify the language. (It defaults to the current system language.) Supports variable expansion. For more information, see Variable Expansion (page 330).

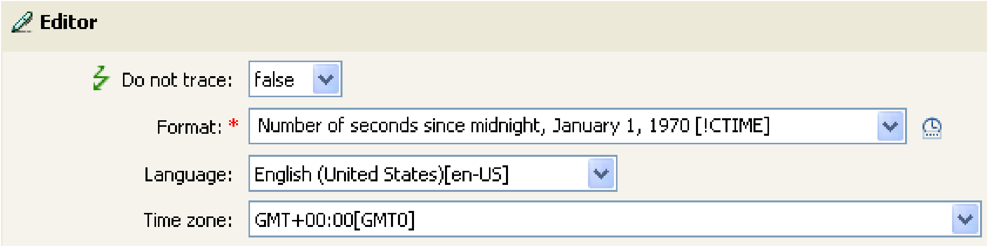
Time zone

Specify the time zone. (It defaults to the current system time zone.) Supports variable expansion. For more information, see Variable Expansion (page 330).

Example



Time(format="!CTIME", lang="en-US", tz="GMT0")



Do not trace: false

Format: * Number of seconds since midnight, January 1, 1970 [!CTIME]

Language: English (United States)[en-US]

Time zone: GMT+00:00[GMT0]

Text

Expands to the text.

Fields

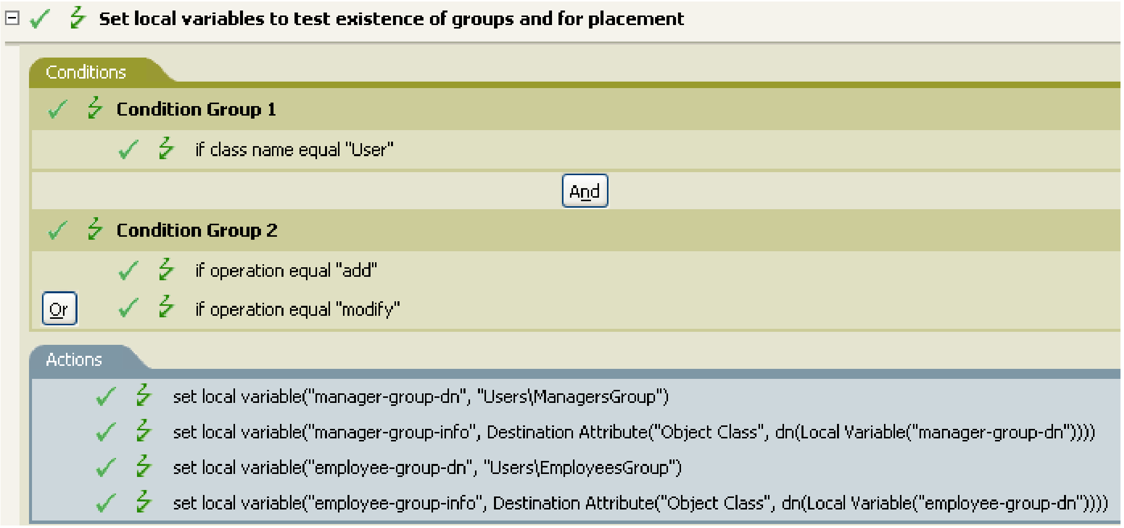
Text

Specify the text. Supports variable expansion. For more information, see Variable Expansion (page 330).

Example

The example is from the Govern Groups for User Based on Title policy, which is available for download from the Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html).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.



Set local variables to test existence of groups and for placement

Conditions

Condition Group 1

if class name equal "User"

And

Condition Group 2

if operation equal "add"

Or if operation equal "modify"

Actions

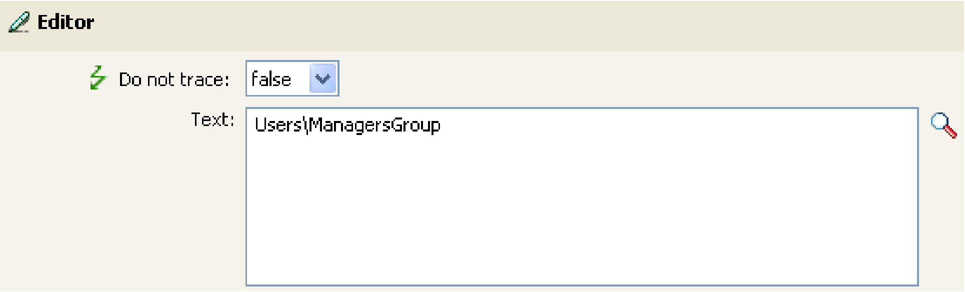
set local variable("manager-group-dn", "Users\ManagersGroup")

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"))))

"Users\ManagersGroup"



Editor

Do not trace: false

Text: Users\ManagersGroup

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.

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.

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 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:

- always
- never
- fallback

Counters Pattern

Select which pattern to use the counter with. The options are:

- first
- last
- all

Start

The starting value of the counter.

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.

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, shutdown driver

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 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.

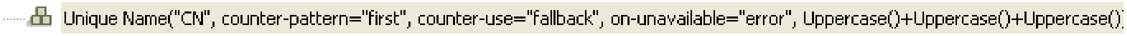
As soon as a proposed name is determined to be unique, the testing of names is stopped and the unique name is returned.

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.

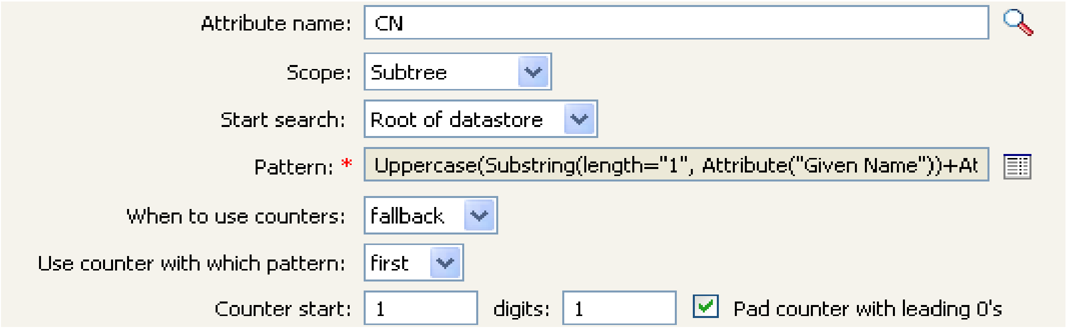
If all specified combinations of patterns and counters are exhausted, then the action specified by the on-unavailable is taken.

Example



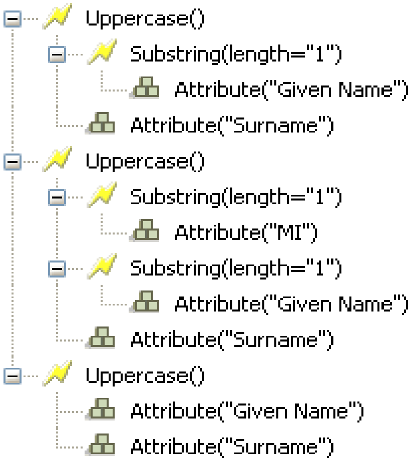
Unique Name("CN", counter-pattern="first", counter-use="fallback", on-unavailable="error", Uppercase()+Uppercase()+Uppercase()

The following is an example of the Editor pane when constructing the unique name argument:



Attribute name:	CN
Scope:	Subtree
Start search:	Root of datastore
Pattern: *	Uppercase(Substring(length="1", Attribute("Given Name"))+At
When to use counters:	fallback
Use counter with which pattern:	first
Counter start:	1
digits:	1
☑	Pad counter with leading 0's

The following pattern was constructed to provide unique names:



```
Uppercase()
    Substring(length="1")
        Attribute("Given Name")
    Attribute("Surname")
Uppercase()
    Substring(length="1")
        Attribute("MI")
    Substring(length="1")
        Attribute("Given Name")
    Attribute("Surname")
Uppercase()
    Attribute("Given Name")
    Attribute("Surname")
```

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.

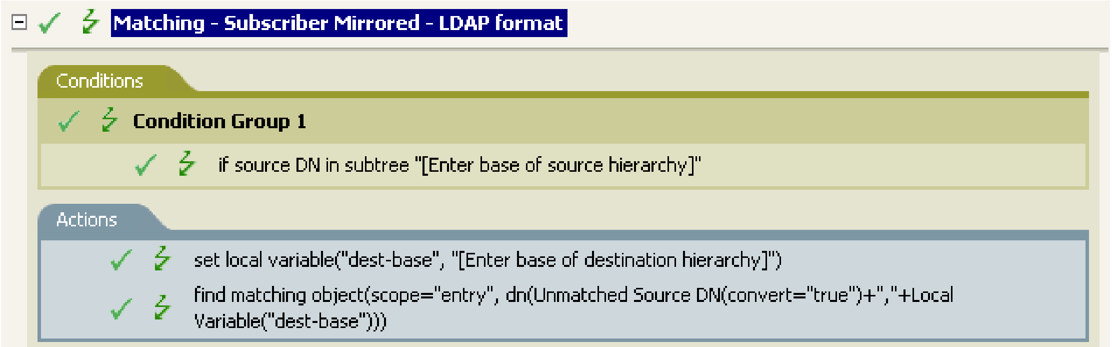
Remarks

If there are no matches, the entire DN is used.

Example

The example is from the predefined rules that come with Identity Manager. For more information, see Matching - Subscriber Mirrored - LDAP Format (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prmatchsubmirror.html#prmatchsubmirror). 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.



Matching - Subscriber Mirrored - LDAP format

Conditions

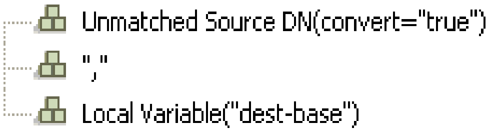
Condition Group 1

if source DN in subtree "[Enter base of source hierarchy]"

Actions

set local variable("dest-base", "[Enter base of destination hierarchy]")

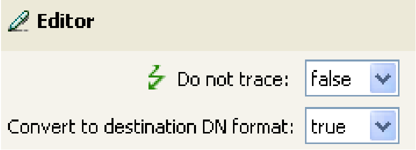
find matching object(scope="entry", dn(Unmatched Source DN(convert="true")+","+Local Variable("dest-base")))



Unmatched Source DN(convert="true")

","

Local Variable("dest-base")



Editor

Do not trace: false

Convert to destination DN format: true

XPath

Expands to results of evaluating an XPath 1.0 expression.

Fields

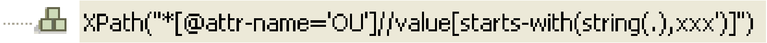
Expression

XPath 1.0 expression to evaluate.

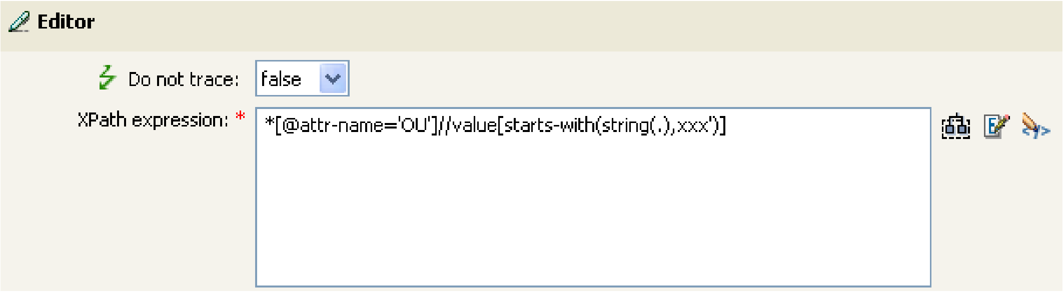
Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Example



```
XPath("*[@attr-name='OU']//value[starts-with(string(.),xxx')]")
```



Do not trace: false

XPath expression: *

```
*[@attr-name='OU']//value[starts-with(string(.),xxx')]
```

Variable Expansion

Allows for the use of dynamic variables in the noun token

Remark

Many noun tokens 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 or global variable with the given name. $<variable-name>$ must be legal variable name. For information on what is 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, it should be escaped with an additional $ (for example, You owe me $$100.00).

15 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.

Base64 Decode

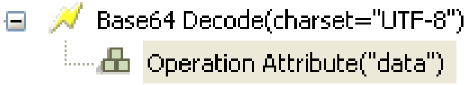
Decodes the result of the enclosed tokens from Base64-encoded data to bytes and then converts the bytes into a string using the specified character set.

Fields

Character Set

Specify the character set that converts the decoded bytes to a string. It can be any Java supported character set. 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 Expansion (page 351).

Example



```
Base64 Decode(charset="UTF-8")
    Operation Attribute("data")
```

Base64 Encode

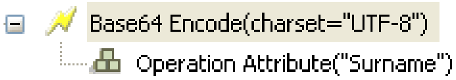
Converts the result of the enclosed tokens to bytes using the specified character set, and then Base64-encodes the bytes.

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 Expansion (page 351).

Example



```
Base64 Encode(charset="UTF-8")
    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 Expansion (page 351).

Source Language

Specify the source language (defaults to the current system language). Supports variable expansion. For more information, see Variable Expansion (page 351).

Source Time Zone

Specify the source time zone (defaults to the current system time zone). Supports variable expansion. For more information, see Variable Expansion (page 351).

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 Expansion (page 351).

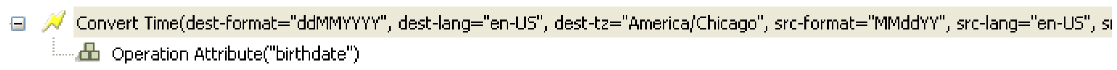
Destination Language

Specify the destination language (defaults to the current system language). Supports variable expansion. For more information, see Variable Expansion (page 351).

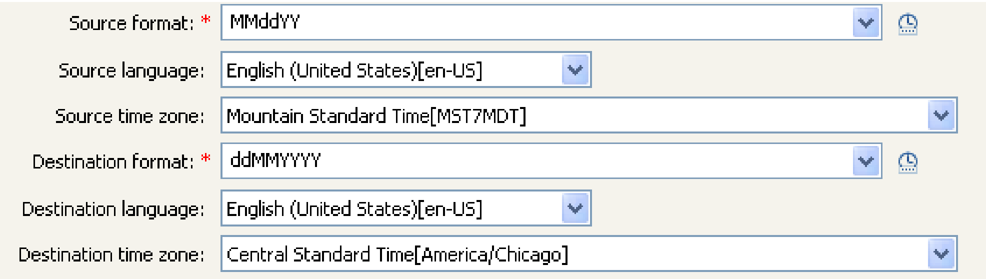
Destination Time Zone

Specify the destination time zone (defaults to the current system time zone). Supports variable expansion. For more information, see Variable Expansion (page 351).

Example



```
Convert Time(dest-format="ddMMYYYY", dest-lang="en-US", dest-tz="America/Chicago", src-format="MMddYY", src-lang="en-US", s
    Operation Attribute("birthdate")
```



Source format: *	MMddYY
Source language:	English (United States)[en-US]
Source time zone:	Mountain Standard Time[MST7MDT]
Destination format: *	ddMMYYYY
Destination language:	English (United States)[en-US]
Destination time zone:	Central Standard Time[America/Chicago]

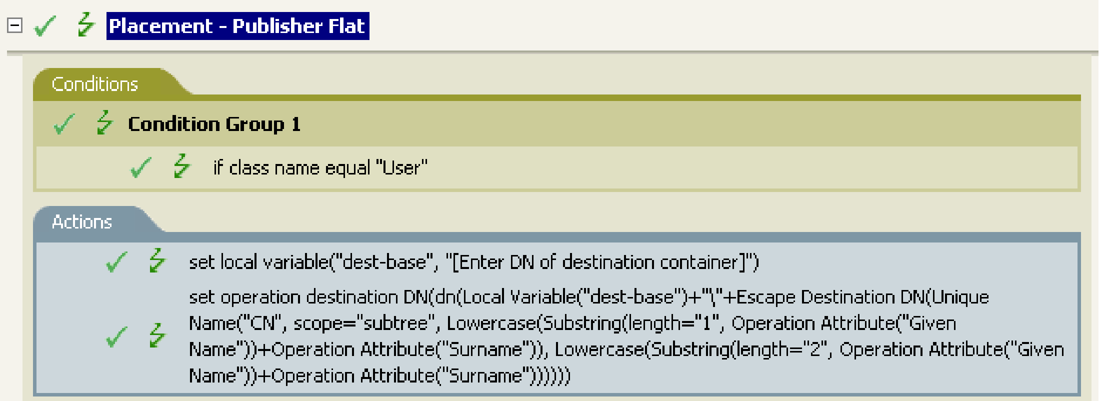
Escape Destination DN

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 Identity Manager. For more information, see Placement - Publisher Flat (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prplacepubflat.html#prplacepubflat). To view the policy in XML, see predef_place_pub_flat.xml (../sample/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.



Placement - Publisher Flat

Conditions

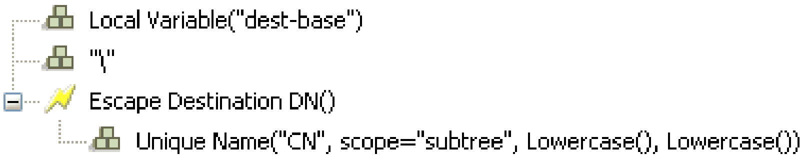
Condition Group 1

if class name equal "User"

Actions

set local variable("dest-base", "[Enter DN of destination container]")

set operation destination DN(dn(Local Variable("dest-base")+"\"+Escape Destination DN(Unique Name("CN", scope="subtree", Lowercase(Substring(length="1", Operation Attribute("Given Name"))+Operation Attribute("Surname")), Lowercase(Substring(length="2", Operation Attribute("Given Name"))+Operation Attribute("Surname"))))))



Local Variable("dest-base")

"\"

Escape Destination DN()

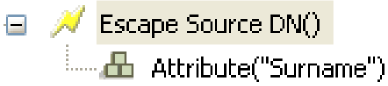
Unique Name("CN", scope="subtree", Lowercase(), Lowercase())

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



```
Escape Source DN()
    Attribute("Surname")
```

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

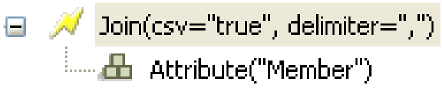
(Optional) Specify the string used to delimit the joined values. Supports variable expansion. For more information, see Variable Expansion (page 351).

Apply CSV Quoting Rules

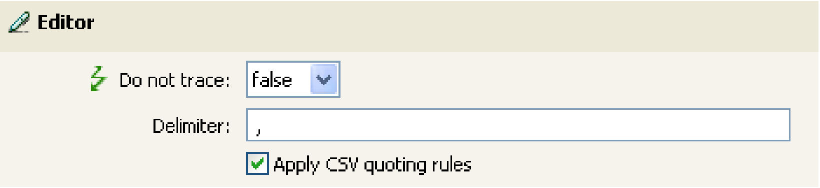
Applies CSV quoting values.

Example

The example combines all of the members of the group into a CSV record.



```
Join(csv="true", delimiter=",")
    Attribute("Member")
```



Do not trace: false

Delimiter: ,

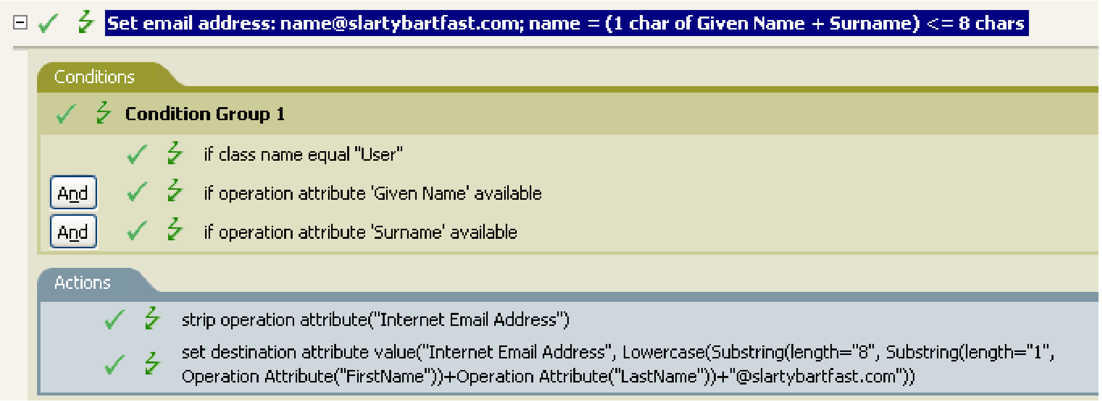
☑ Apply CSV quoting rules

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Command-SetEmailByGivenNameAndSurnam.xml (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).



Set email address: name@slartybartfast.com; name = (1 char of Given Name + Surname) <= 8 chars

Conditions

Condition Group 1

if class name equal "User"

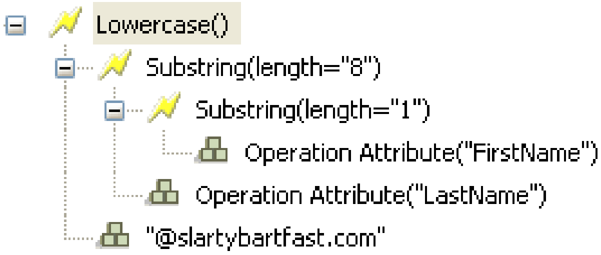
And if operation attribute 'Given Name' available

And if operation attribute 'Surname' available

Actions

strip operation attribute("Internet Email Address")

set destination attribute value("Internet Email Address", Lowercase(Substring(length="8", Substring(length="1", Operation Attribute("FirstName"))+Operation Attribute("LastName"))+"@slartybartfast.com"))



```
Lowercase()
    Substring(length="8")
        Substring(length="1")
            Operation Attribute("FirstName")
        Operation Attribute("LastName")
    "@slartybartfast.com"
```

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. For more information, see Variable Expansion (page 351).

Render Browse DN Relative to Policy

When it is enabled, it displays the mapping table DN relative to the policy. This is the default.

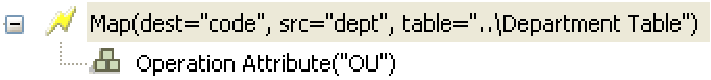
Source Column Name

Specify the name of the source column. Supports variable expansion. For more information, see Variable Expansion (page 351).

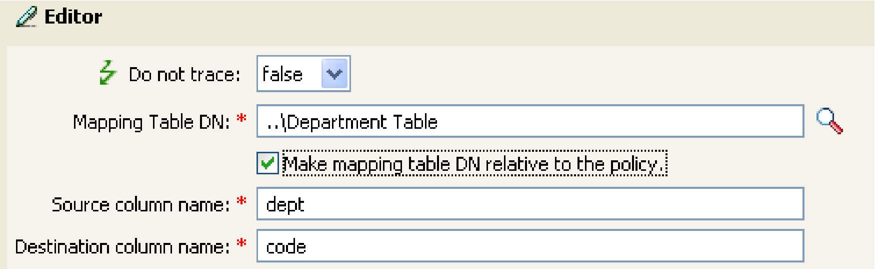
Destination Column Name

Specify the name of the destination column. Supports variable expansion. For more information, see Variable Expansion (page 351).

Example



```
Map(dest="code", src="dept", table="..\Department Table")
    Operation Attribute("OU")
```



Editor

Do not trace: false

Mapping Table DN: * ..\Department Table

☑ Make mapping table DN relative to the policy.

Source column name: * dept

Destination column name: * code

Parse DN

Converts the enclosed token's DN to an alternate format.

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 eDirectory: 0xfeff, 0xfffe, 0xfffd, and 0xffff.
- Relative RDN Delimiter
- RDN Delimiter
- Name Divider
- Name Value Delimiter
- Wildcard Character

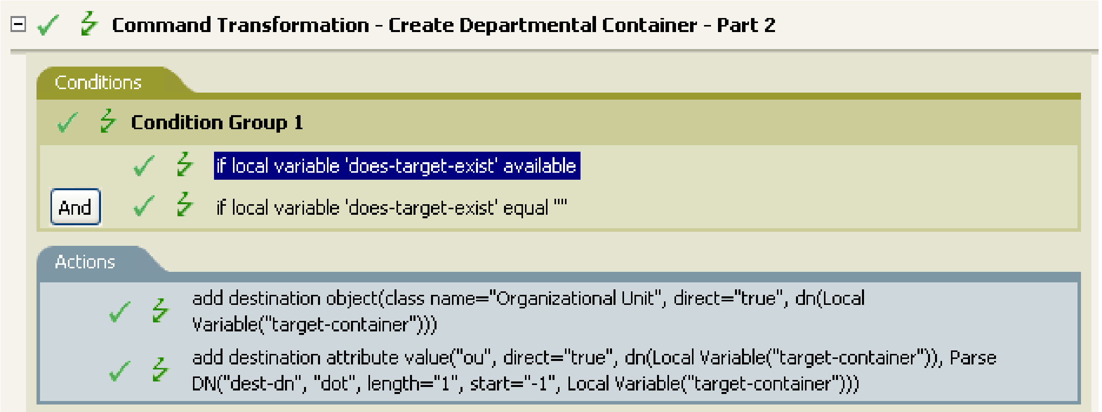
- Escape Character

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 Identity Manager. For more information, see Command Transformation - Create Departmental Container - Part 1 and Part 2 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prdeptcontainer.html#prdeptcontainer). To view the policy in XML, see predef_command_create_dept_container2.xml (../sample/predef_command_create_dept_container2.xml).



Command Transformation - Create Departmental Container - Part 2

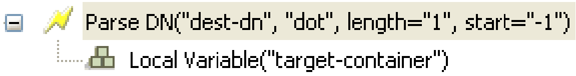
Condition Group 1

if local variable 'does-target-exist' available

And if local variable 'does-target-exist' equal ""

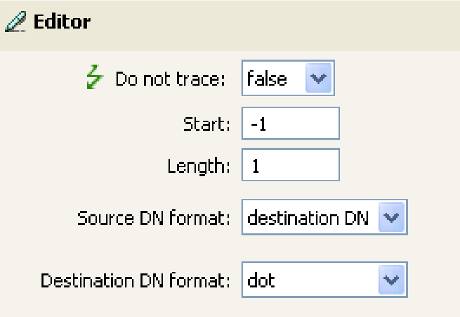
add destination object(class name="Organizational Unit", direct="true", dn(Local Variable("target-container")))

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")))



Parse DN("dest-dn", "dot", length="1", start="-1")

Local Variable("target-container")



Editor

Do not trace: false

Start: -1

Length: 1

Source DN format: destination DN

Destination DN format: dot

The Parse DN token is taking the information from the source DN and converting it to the 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 Expansion (page 351).

Replace With

Specify the replacement string. Supports variable expansion. For more information, see Variable Expansion (page 351).

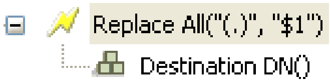
Remarks

For details on creating regular expressions, see:

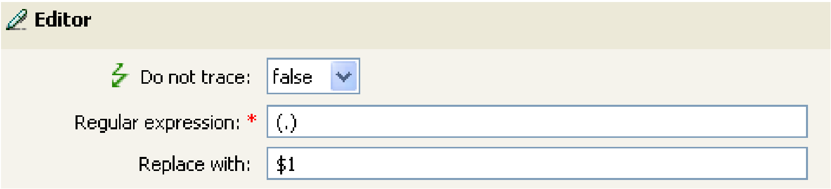
- Sun's Java Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html)
- Sun's Java Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Matcher.html#replaceAll (java.lang.String))

The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.

Example



```
Replace All("(.)", "$1")
    Destination DN()
```



Editor

Do not trace: false

Regular expression: * (.)

Replace with: $1

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 Expansion (page 351).

Replace With

Specify the replacement string. Supports variable expansion. For more information, see Variable Expansion (page 351).

Remarks

The matching instance is replaced by the string specified in the *Replace with field*.

For details on creating regular expressions, see:

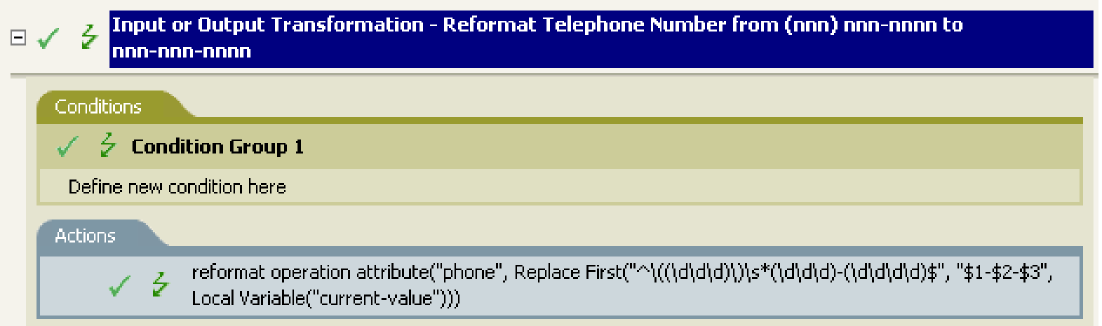
- Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html)
- Sun's Web site (java.lang.String) (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Matcher.html#replaceAll (java.lang.String))

The pattern option CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed 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 Identity Manager. For more information, see Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_designer/data/prreformattel1.html#prreformattel1). To view the policy in XML, see predef_transformation_reformat_telephone1 (../samples/predef_transformation_reformat_telephone1.xml).

The Replace First token is used in the Reformat Operation Attribute action.

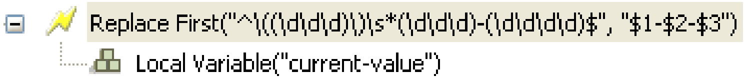


Input or Output Transformation - Reformat Telephone Number from (nnn) nnn-nnnn to nnn-nnn-nnnn

Condition Group 1

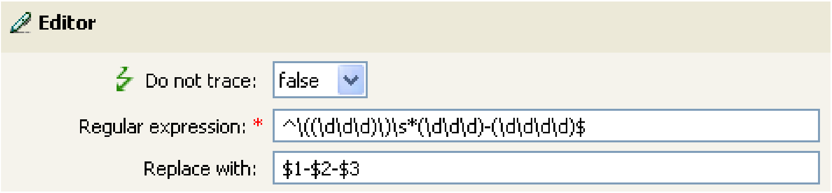
Define new condition here

```
reformat operation attribute("phone", Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3",
Local Variable("current-value")))
```



Replace First("^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$", "$1-$2-$3")

Local Variable("current-value")



Do not trace: false

Regular expression: * ^\((\d\d\d)\)\s*(\d\d\d)-(\d\d\d\d)$

Replace with: $1-$2-$3

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

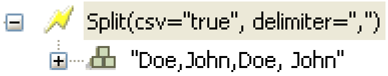
Delimiter

Regular expression that matches the delimiter characters. Supports variable expansion. For more information, see Variable Expansion (page 351).

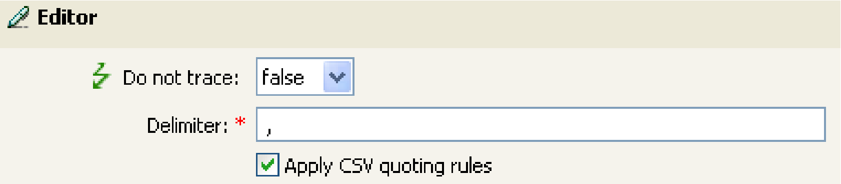
Apply CSV Quoting Rules

Applies CSV quoting values.

Example



```
Split(csv="true", delimiter=",")
    "Doe,John,Doe, John"
```



Editor

Do not trace: false

Delimiter: * ,

☑ Apply CSV quoting rules

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.

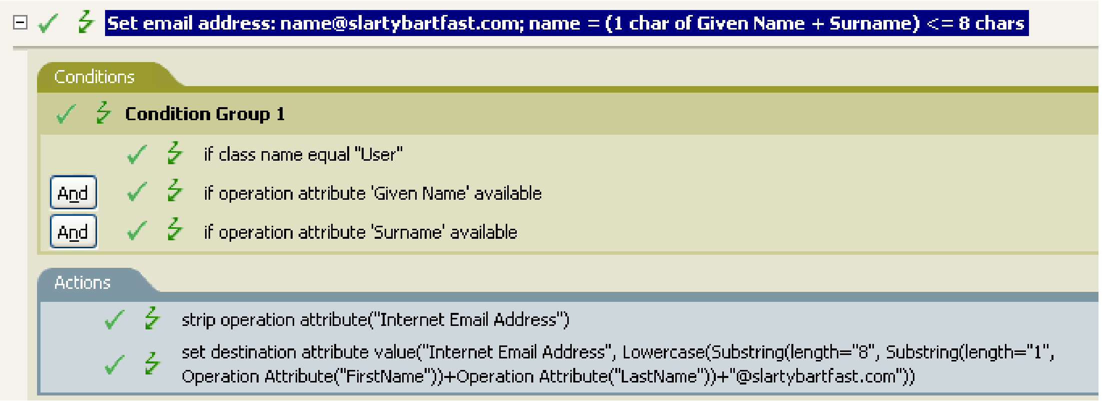
For example, if the start is specified as -2, then it starts reading the first character from the end. If -3 is specified, then is starts 2 characters from the end.

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 or the original string. If -2 is specified, the length is the entire -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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 001-Command-SetEmailByGivenNameAndSurname.xml (../samples/001-Command-SetEmailByGivenNameAndSurname.xml).



Set email address: name@slartybartfast.com; name = (1 char of Given Name + Surname) <= 8 chars

Condition Group 1

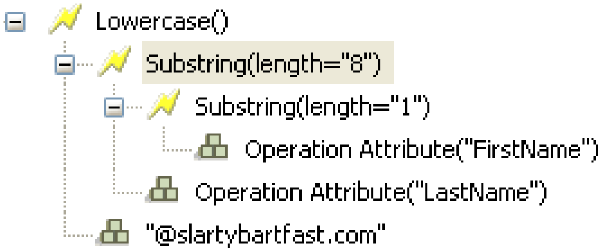
if class name equal "User"

And if operation attribute 'Given Name' available

And if operation attribute 'Surname' available

strip operation attribute("Internet Email Address")

set destination attribute value("Internet Email Address", Lowercase(Substring(length="8", Substring(length="1", Operation Attribute("FirstName"))+Operation Attribute("LastName"))+"@slartybartfast.com"))



```
Lowercase()
    Substring(length="8")
        Substring(length="1")
            Operation Attribute("FirstName")
        Operation Attribute("LastName")
    "@slartybartfast.com"
```

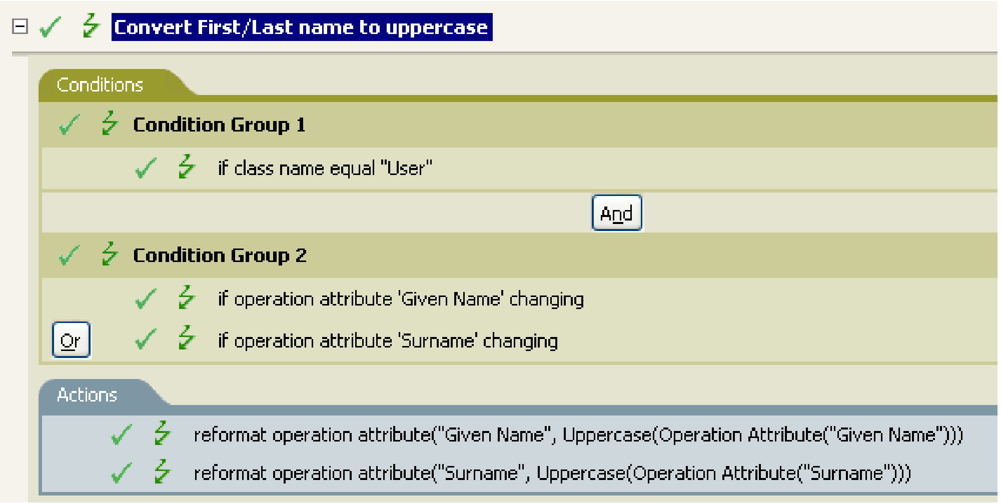
The Substring token is used twice in the action Set Destination Attribute Value. It takes the first character of the First Name attribute and adds eight characters of the Last Name attribute together to form one substring.

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 Novell Support Web site. For more information, see Downloading Identity Manager Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policydownloadable.html). To view the policy in XML, see 002-Command-UppercaseNames.xml (../samples/002-Command-UppercaseNames.xml).



Convert First/Last name to uppercase

Conditions

Condition Group 1

if class name equal "User"

And

Condition Group 2

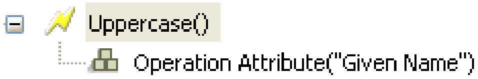
if operation attribute 'Given Name' changing

Or if operation attribute 'Surname' changing

Actions

reformat operation attribute("Given Name", Uppercase(Operation Attribute("Given Name")))

reformat operation attribute("Surname", Uppercase(Operation Attribute("Surname")))



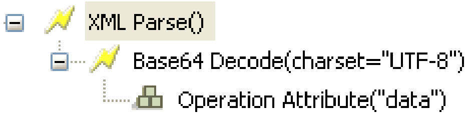
Uppercase()

Operation Attribute("Given Name")

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

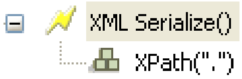


```
XML Parse()
    Base64 Decode(charset="UTF-8")
        Operation Attribute("data")
```

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.

Example



```
XML Serialize()
    XPath(".")
```

Variable Expansion

Allows for the use of dynamic variables in the verb token

Remark

Many verb tokens 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 or global variable with the given name. $<variable-name>$ must be legal variable name. For information on what is 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, it should be escaped with an additional $ (for example, You owe me $$100.00).

Pre-Identity Manager 3.5 Builders

16

Although you define most arguments using the Argument Builder, there are several more builders 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:

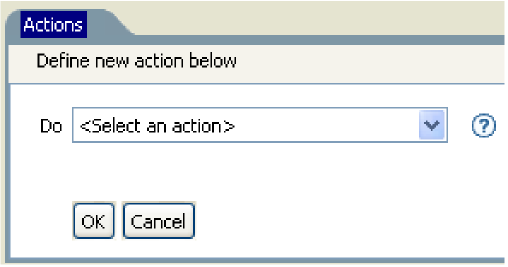
16.1 Action Builder

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

16.1.1 Creating an Action

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

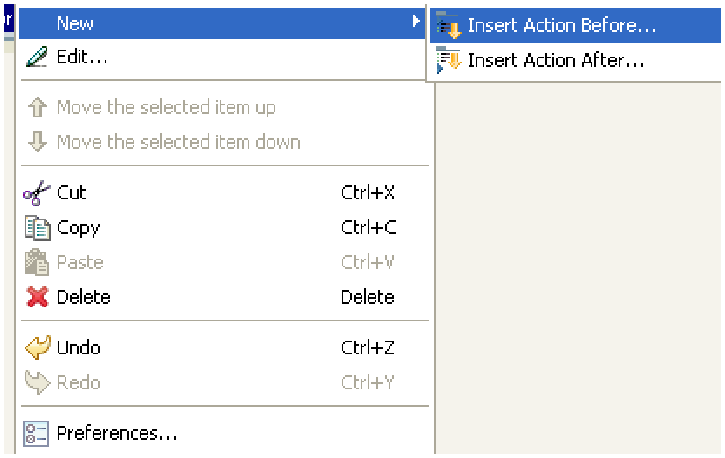
2 Double-click the *Actions* tab to launch the Action Builder.



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

16.1.2 Additional Options for the Action Builder

1 Right-click the action to see the additional options:



- **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 the selected item up:** Moves the selected action up in the order of execution.
- **Move the selected item down:** Moves the selected action down in the order of execution.
- **Cut, Copy, Paste, or Delete an Action:** Cuts, copies, pastes, or deletes the action.
- **Undo or Redo:** Undoes or redoes the last action.
- **Preferences:** Allows you to set default functionality in the Policy Builder.
- **Help:** Select an action, then click the *Help* icon to see information specific to that action.

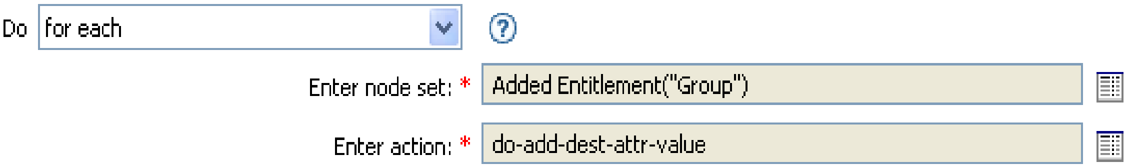
16.2 Actions Builder

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 arguments* icon .

- For Each (page 416)
- Implement Entitlement (page 419)

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 16-1** For Each Action*



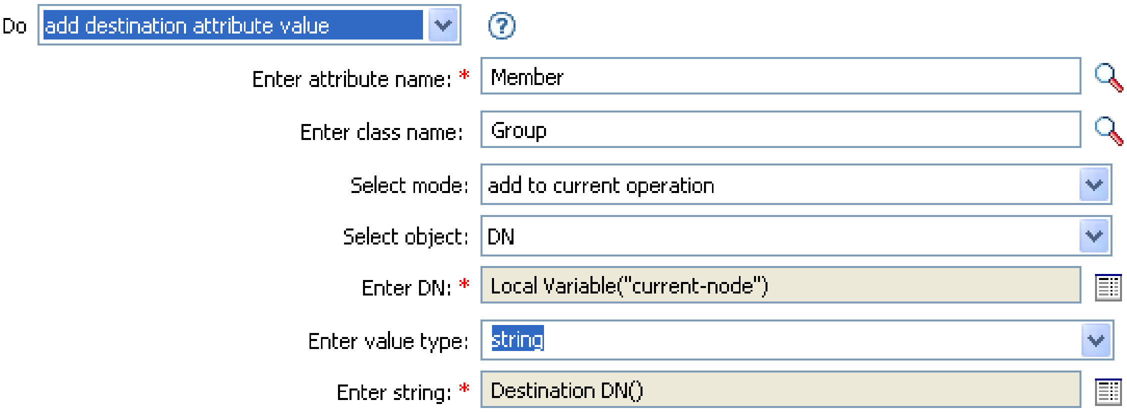
Do: for each

Enter node set: * Added Entitlement("Group")

Enter action: * do-add-dest-attr-value

To define the action of the add destination attribute value, click the icon that 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 16-2 *Actions Builder*



Do: add destination attribute value

Enter attribute name: * Member

Enter class name: Group

Select mode: add to current operation

Select object: DN

Enter DN: * Local Variable("current-node")

Enter value type: string

Enter string: * Destination DN()

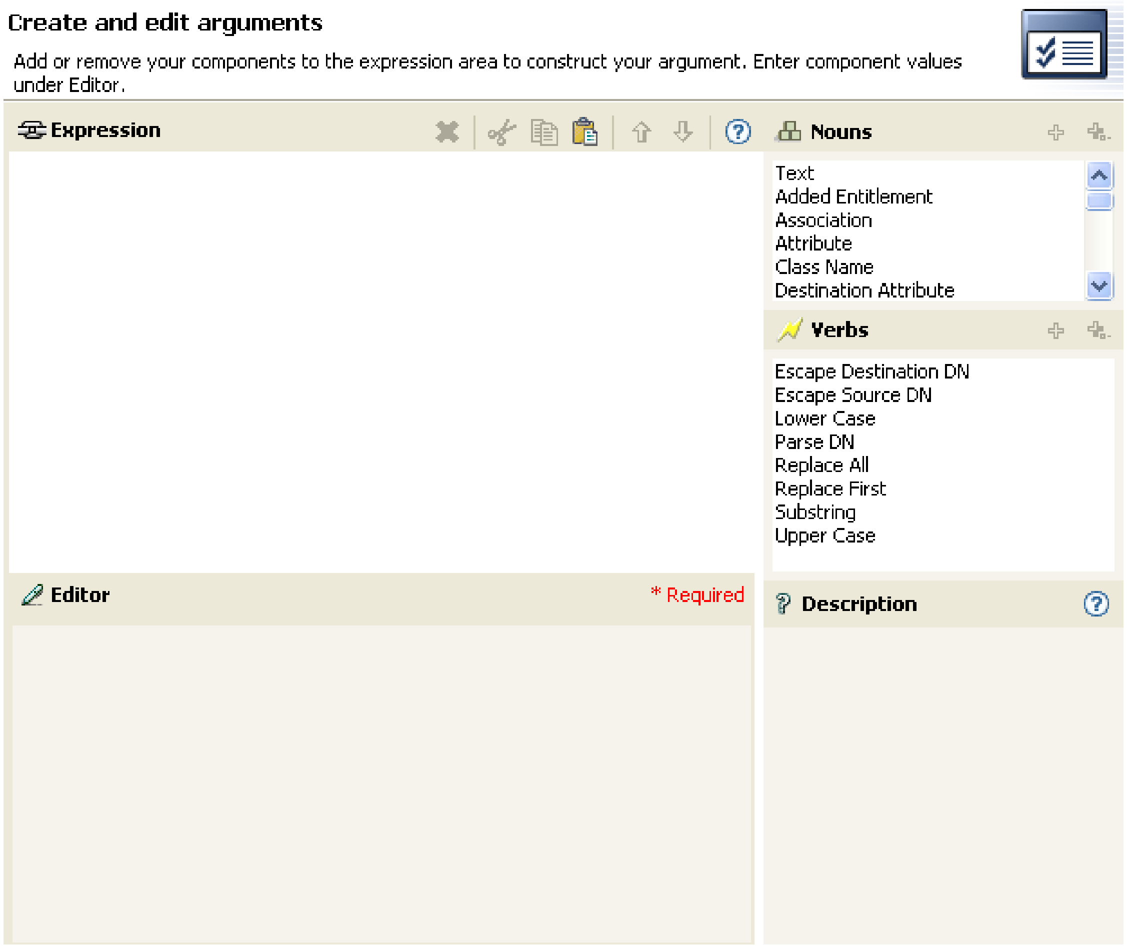
16.3 Argument Builder

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

The Argument Builder consists of five separate sections:

- **Nouns:** Contains a list of all of the available noun tokens. Select a noun token, then click *Add* to add the noun token to the *Expression* pane. See "Pre-Identity Manager 3.5 Noun Tokens" on page 453 for more information.
- **Verbs:** Contains a list of all of the available verb tokens. Select a verb token, then click *Add* to add the verb token to the *Expression* pane. See "Pre-Identity Manager 3.5 Verb Tokens" on page 479 for more information.
- **Description:** Contains a brief description of the 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.

Figure 16-3 Pre-Identity Manager 3.5 Argument Builder



Create and edit arguments

Add or remove your components to the expression area to construct your argument. Enter component values under Editor.

- Section 16.3.1, “Launching the Argument Builder,” on page 356
- Section 16.3.2, “Argument Builder Example,” on page 358

16.3.1 Launching the Argument Builder

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

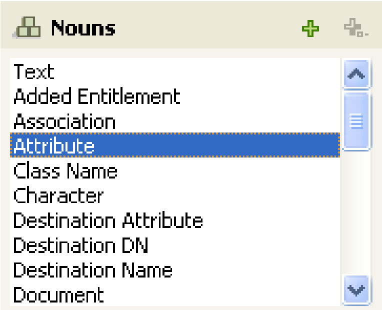
- Add Association (page 399)
- Add Destination Attribute Value (page 400)
- Add Destination Object (page 401)
- Add Source Attribute Value (page 402)
- Append XML Text (page 405)
- Clear Destination Attribute Value (page 407) (when the selected object is DN or Association)
- Clear Source Attribute Value (page 409) (when the selected object is DN or Association)
- Delete Destination Object (page 413) (when the selected object is DN or Association)
- Delete Source Object (page 414) (when the selected object is DN or Association)

- Find Matching Object (page 415)
- For Each (page 416)
- Move Destination Object (page 420)
- Move Source Object (page 421)
- Reformat Operation Attribute Value (page 422)
- Remove Association (page 423)
- Remove Destination Attribute Value (page 424)
- Remove Source Attribute Value (page 425)
- Rename Destination Object (page 426) (when the selected object is DN or Association and Enter String)
- Rename Source Object (page 428) (when the selected object is DN or Association and Enter String)
- Set Destination Attribute Value (page 432) (when the selected object is DN or Association and Enter Value Type is not structured)
- Set Destination Password (page 433)
- Set Local Variable (page 434)
- Set Operation Association (page 435)
- Set Operation Class Name (page 436)
- Set Operation Destination DN (page 437)
- Set Operation Property (page 438)
- Set Operation Source DN (page 439)
- Set Operation Template DN (page 440)
- Set Source Attribute Value (page 441)
- Set Source Password (page 442)
- Set XML Attribute (page 445)
- Status (page 446)
- Trace Message (page 449)

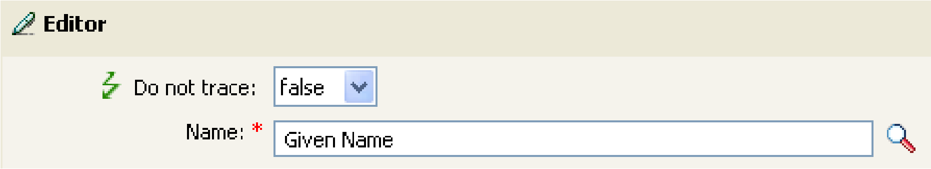
16.3.2 Argument Builder Example

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

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



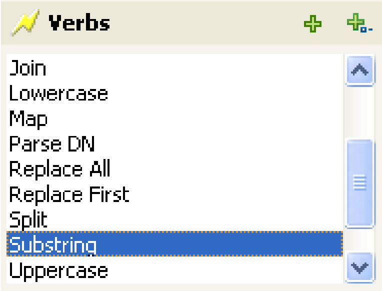
2 Specify or select the Given Name attribute.


Editor

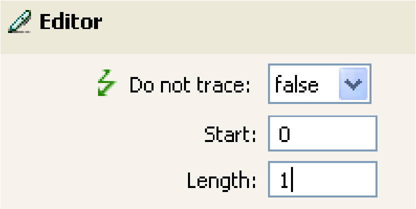
Do not trace: false

Name: * Given Name

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



4 Type 1 in the *Length* field.

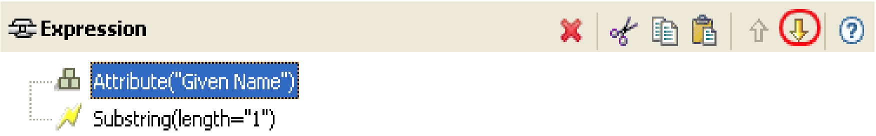

Editor

Do not trace: false

Start: 0

Length: 1

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



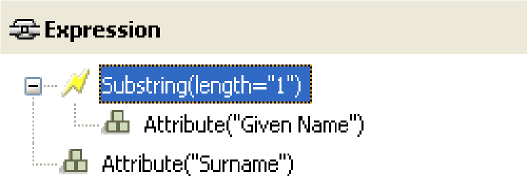
Expression

Attribute("Given Name")

Substring(length="1")

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

7 Specify or browse to the *Surname* attribute.



Expression

Substring(length="1")

Attribute("Given Name")

Attribute("Surname")

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

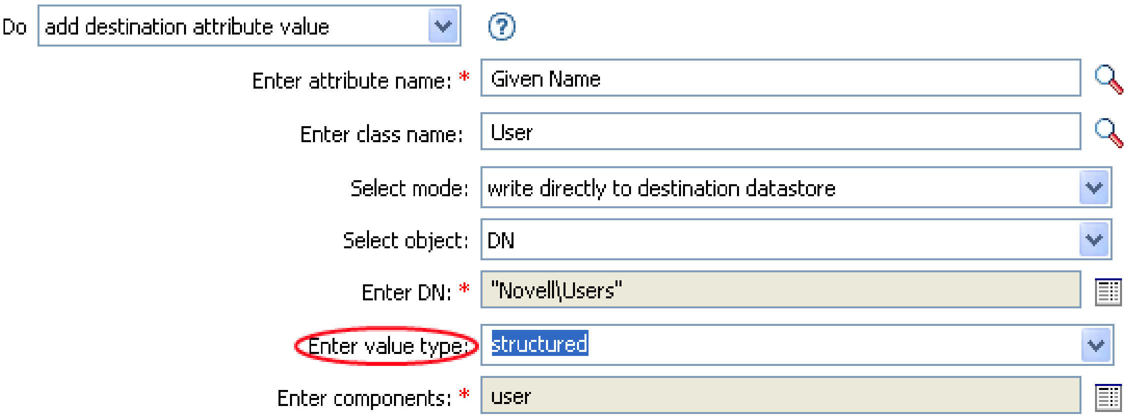
8 Click *OK* to save the argument.

16.4 Action Argument Component Builder

To launch the Action Argument Component Builder, select one of the following actions when the *Enter value type* selection is *structured*, then click the *Edit components* icon ▤.

- Add Destination Attribute Value (page 400)
- Add Source Attribute Value (page 402)
- Reformat Operation Attribute Value (page 422)
- Remove Destination Attribute Value (page 424)
- Remove Source Attribute Value (page 425)
- Set Destination Attribute Value (page 432)
- Set Source Attribute Value (page 441)

Figure 16-4 *Add Destination Attribute Value Action*



Do: add destination attribute value

Enter attribute name: * Given Name

Enter class name: User

Select mode: write directly to destination datastore

Select object: DN

Enter DN: * "Novell\Users"

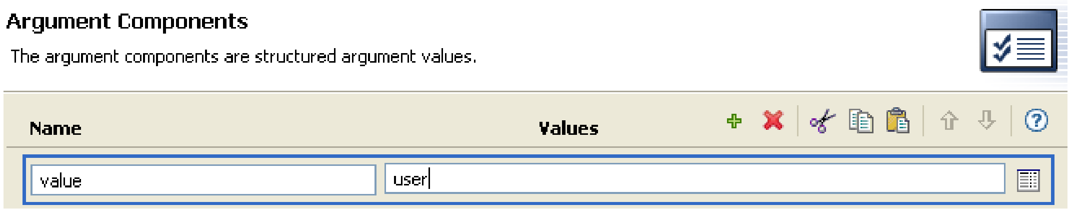
Enter value type: structured

Enter components: * user

1 Click the *Edit the components* icon ▤ when the value type is set to structured.

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.



Argument Components

The argument components are structured argument values.

Name	Values
value	user

3 Click *Finish*.

16.5 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. Condition structures define the logic of condition groups. The two condition structures are:

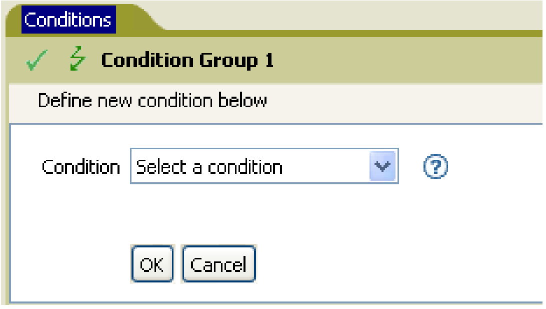
- *OR Conditions, AND Groups*
- *AND Conditions, OR Groups*

- Section 16.5.1, "Creating a Condition," on page 360
- Section 16.5.2, "Additional Options for the Condition Builder," on page 361

16.5.1 Creating a Condition

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

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



Conditions

Condition Group 1

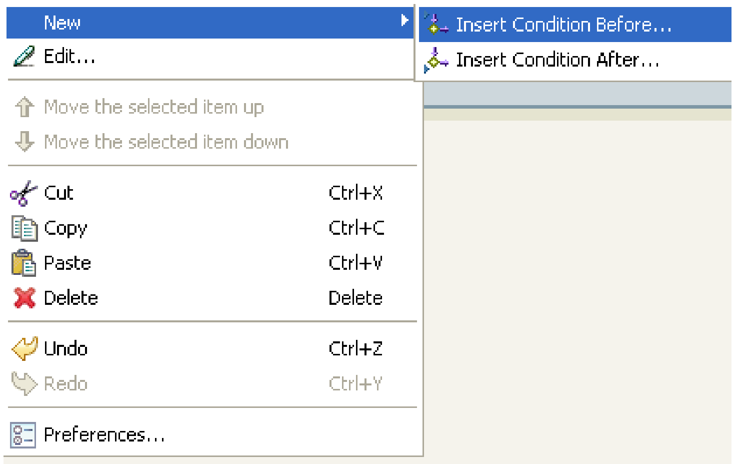
Define new condition below

Condition: Select a condition

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

16.5.2 Additional Options for the Condition Builder

1 Right-click the condition to see the additional options:


New
Edit...
Move the selected item up
Move the selected item down
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Undo Ctrl+Z
Redo Ctrl+Y
Preferences...
Insert Condition Before...
Insert Condition After...

- **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 the selected item up:** Moves the selected condition up in the order of execution.
- **Move the selected item down:** Moves the selected condition down in the order of execution.
- **Cut, Copy, Paste, or Delete:** Cuts, copies, pastes, or deletes the condition.
- **Undo or Redo:** Undoes or redoes the last action.
- **Preferences:** Allows you to set default functionary in the Policy builder.
- **Help:** Select a condition, then click the *Help* icon to see information specific to that condition.

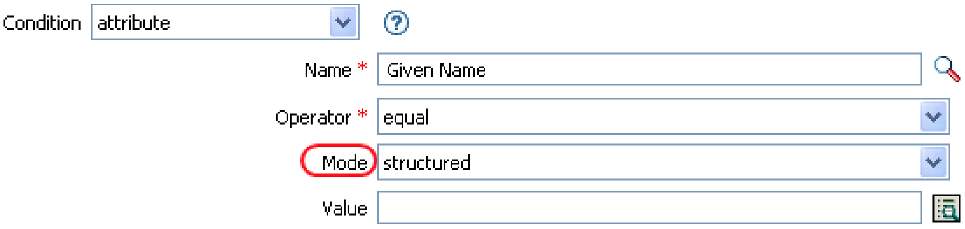
For additional information on the Condition Builder and the rules, see Section 3.4, “Creating a Rule,” on page 32.

16.6 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 371)
- If Destination Attribute (page 375)
- If Association (page 370)

Figure 16-5 *If Attribute mode*



Condition: attribute

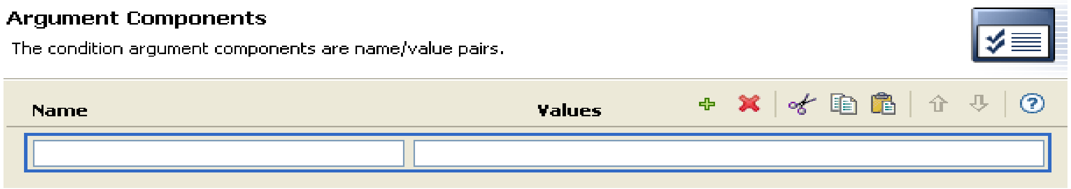
Name *: Given Name

Operator *: equal

Mode: structured

Value:

1 Specify the name and value of the condition component.



Argument Components

The condition argument components are name/value pairs.

Name	Values

2 Click *Finish*.

16.7 Match Attribute Builder

The Match Attribute Builder enables you to select attributes and values used by the Find Matching Object (page 415) action to determine if a matching object exists in a data store.

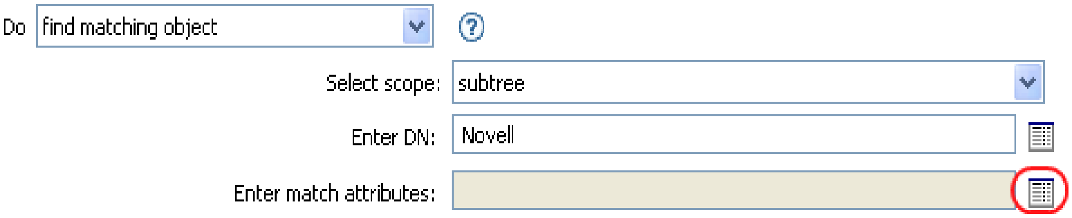
For example, if you wanted to match users based on a common name and a location:

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.



Do: find matching object

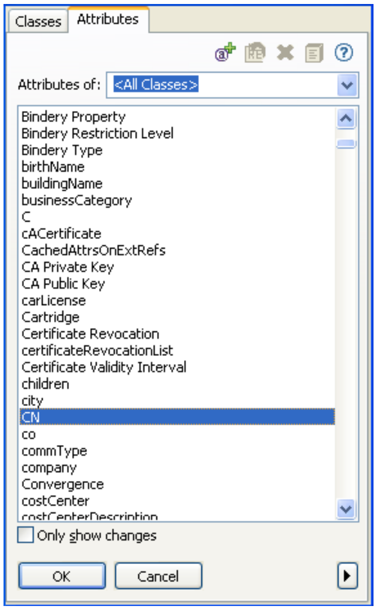
Select scope: subtree

Enter DN: Novell

Enter match attributes:

5 Click the *Browse attributes* icon to launch the Schema Browser.

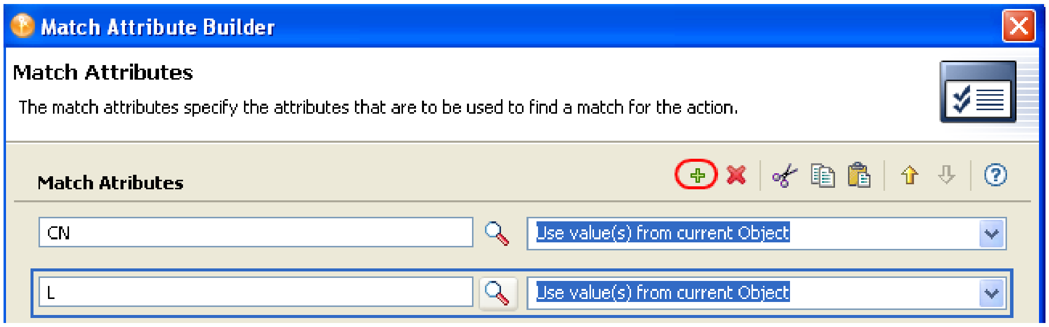
6 Click the *Attributes* tab, then browse to and select the desired attribute.



Attributes of: <All Classes>

7 Click *OK*.

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



The match attributes specify the attributes that are to be used to find a match for the action.

CN — Use value(s) from current Object

L — Use value(s) from current Object

8 Click *Finish*.

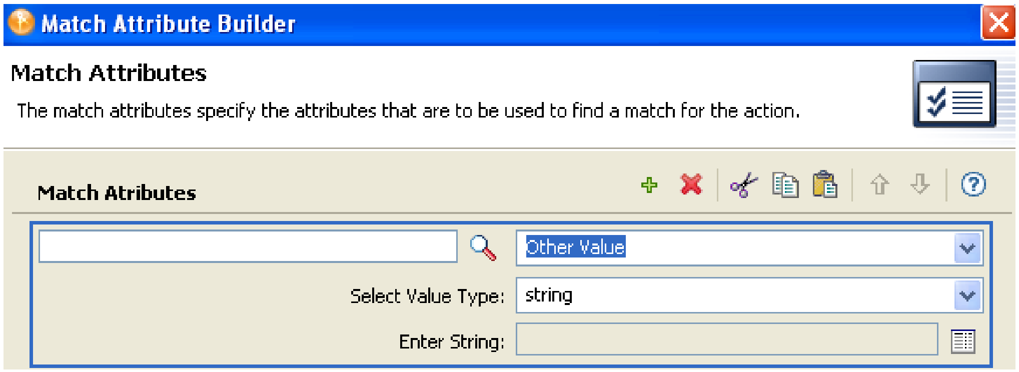
The Match Attribute Builder also allows you to specify another value, instead of using the value from the current object. Select *Other Value* instead of *Use values from current object*, to use a different value. There are multiple value types to specify:

- counter
- dn
- int
- interval
- octet
- state
- structured

- teleNumber
- time

To use the *Other Value*:

1 Launch the Match Attribute Builder, then select *Other Value*.



Match Attribute Builder

Match Attributes

The match attributes specify the attributes that are to be used to find a match for the action.

Match Atributes

Other Value

Select Value Type: string

Enter String:

2 Select the desired value type.

3 Specify the value, then click *OK*.

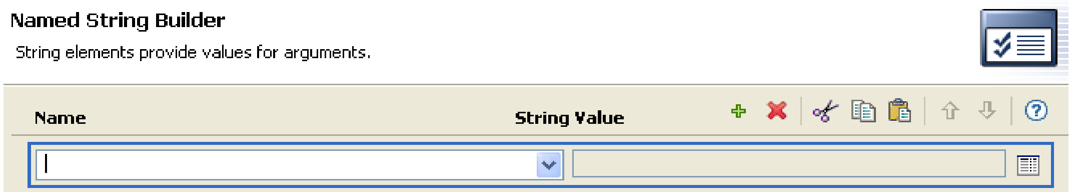
16.8 Named String Builder

To launch the Named String Builder, select one of the following actions, then click the *Edit the strings* icon ▤.

- Generate Event (page 417)
- Send Email (page 429)
- Send Email from Template (page 430)

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.



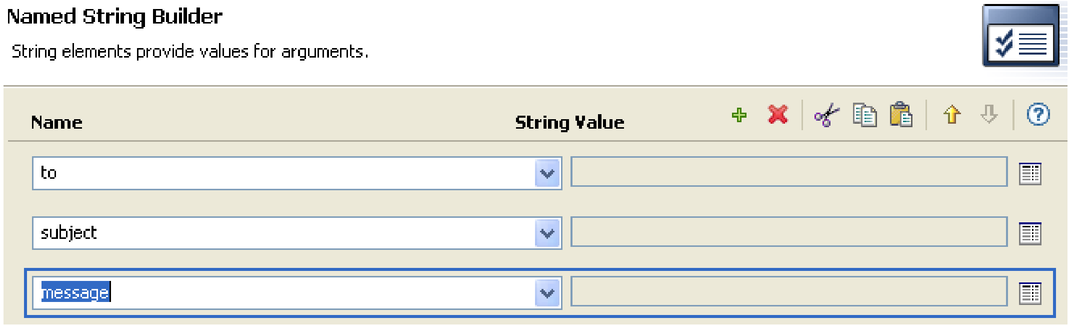
Named String Builder

String elements provide values for arguments.

Name	String Value

3 Click *Finish*.

For a Send Email action, the named strings correspond to the elements of the e-mail:



Named String Builder

String elements provide values for arguments.

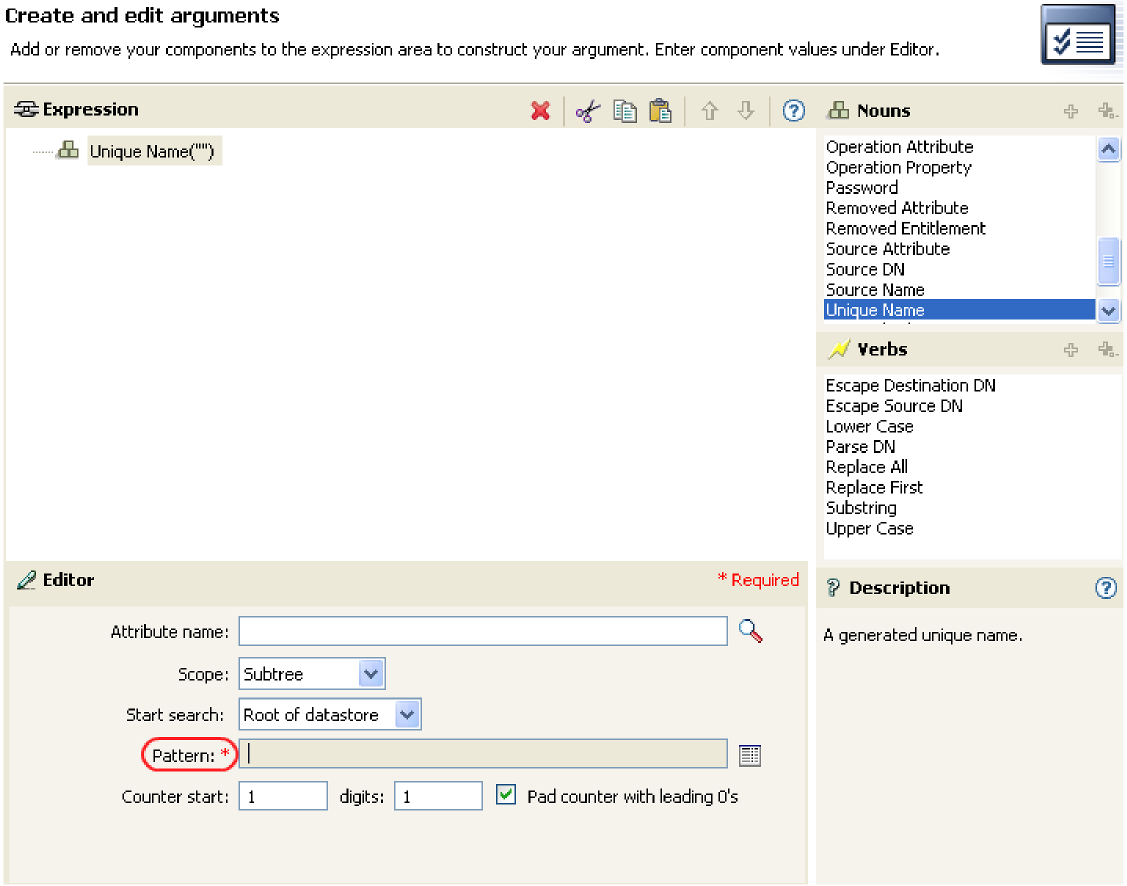
Name	String Value
to	
subject	
message	

A complete list of possible values is contained in the help file corresponding to the action that launches the Named String Builder.

16.9 Pattern String Builder

You can launch the Pattern String Builder from the Argument Builder editor when the Unique Name (page 475) token is selected. The Argument Builder editor pane shows a Pattern field where you can click to launch the Pattern String Builder.

Figure 16-6 *Unique Name Token in the Argument Builder*



Create and edit arguments

Add or remove your components to the expression area to construct your argument. Enter component values under Editor.

Expression: Unique Name("")

Attribute name:

Scope: Subtree

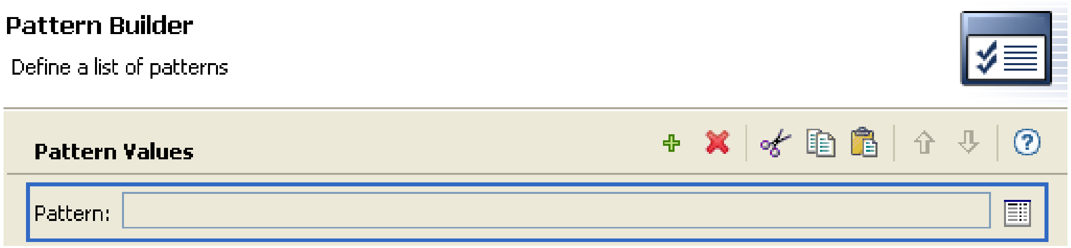
Start search: Root of datastore

Pattern: *

Counter start: 1 digits: 1 ☑ Pad counter with leading 0's

Description: A generated unique name.

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.



Pattern Builder

Define a list of patterns

Pattern Values

Pattern:

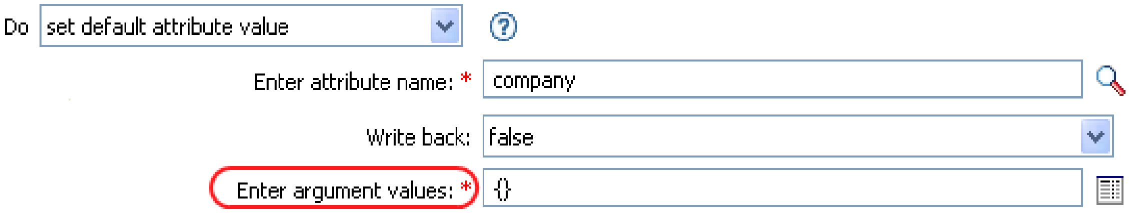
3. Click *Finish*.

16.10 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 431)

Figure 16-7 *Set Default Attribute Value*



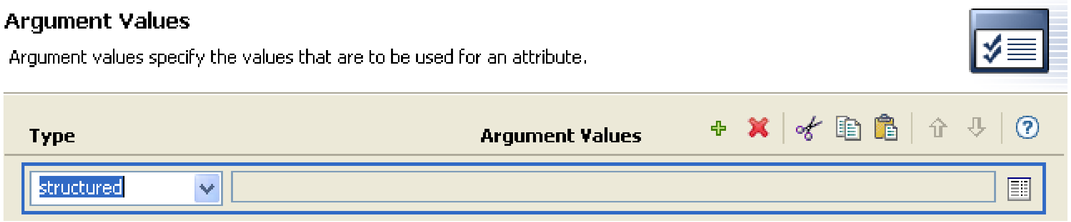
Do: set default attribute value

Enter attribute name: * company

Write back: false

Enter argument values: * {}

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



Argument Values

Argument values specify the values that are to be used for an attribute.

Type	Argument Values
structured	

3. Click the *Edit the arguments* icon.
4. Create the value of the action component.

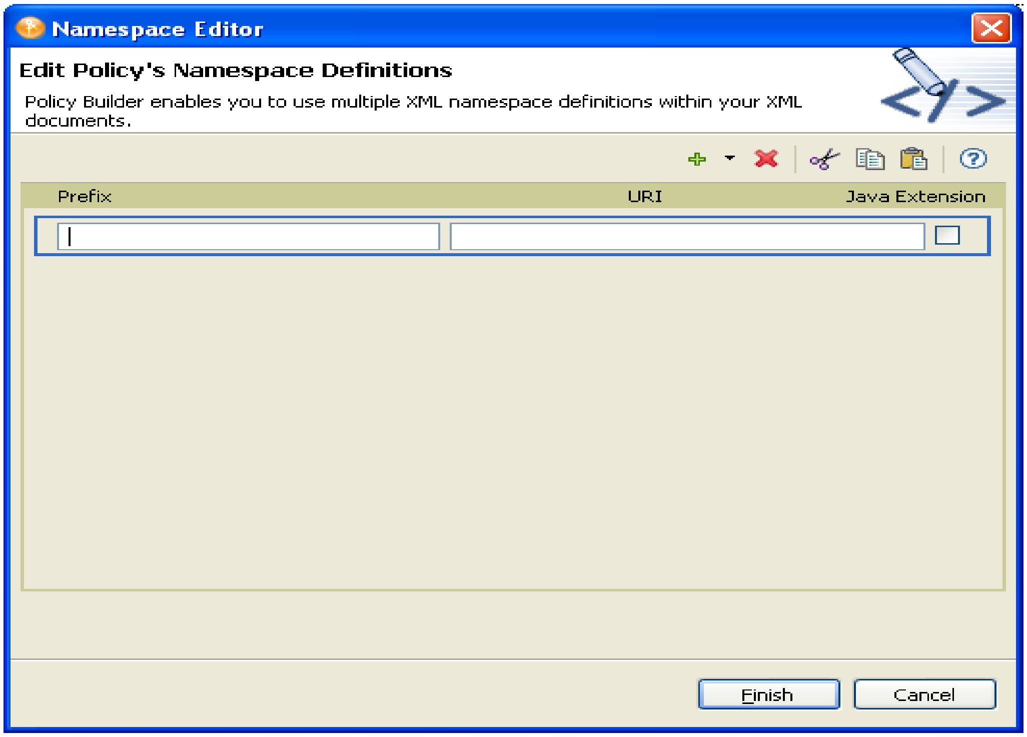
   You can type the value, or click the *Edit the arguments* icon to create the value in the Argument Builder.
5. Click *Finish*.

16.11 Namespace Editor

The Policy Builder enables you to use multiple XML namespaces within your XML documents. To define a namespace, specify the namespace prefix in the *Name* field, and the URI in the *URI* field. Leave the *Java Extension* check box deselected.

You can also access Java* classes through XPath 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.

Figure 16-8 *Namespace Editor*



Namespace Editor

Edit Policy's Namespace Definitions

Policy Builder enables you to use multiple XML namespaces definitions within your XML documents.

Prefix	URI	Java Extension
|		□

16.11.1 Accessing Java Classes Using Namespaces

Novell provides several Identity Manager Java classes that can be called using XPath expressions from the Policy Builder. The following links open Javadoc references for these Java classes:

- com.novell.nds.dirxml.driver.XdsQueryProcessor (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/XdsQueryProcessor.html)
- com.novell.nds.dirxml.driver.XdsCommandProcessor (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/XdsCommandProcessor.html)
- com.novell.nds.dirxml.driver.DNConverter (http://developer.novell.com/documentation/dirxml/dirxmlbk/api/com/novell/nds/dirxml/driver/DNConverter.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 Novell Java classes listed above, consult the DirXML Driver Developer Kit (http://developer.novell.com/documentation/dirxml/dirxmlbk/ref/dirxmlfaq.html).

Pre-Identity Manager 3.5 Conditions

17

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.

This section contains detailed information about all conditions that are available through the pre-Identity Manager 3.5 Policy Builder interface.

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

Select the condition test type.

Operator	Returns True when...
Associated	There is an established association for the current object.
Not Association	There is not an established association for the current object.
Available	There is a non-empty association value specified by the current operation.
Not available	The association is not available for the current object.
Equal	The association value specified by the current operation is exactly equal to the content of the if association.
Not Equal	The association value specified by the current operation is not equal to the content of the if association.

Value

Contains the value defined for the select operator. The value is used by the condition.

- Equal
- Not Equal

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.

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.
Not Available	Available would return False.
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 using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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. See Sun’s Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.

Mode	Description
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

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.
Not Available	Available would return False.
Equal	There is an object class name available in the current operation, and it equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

If Destination Attribute

Performs a test on attribute values of the current object in the destination data store. The test performed depends on the specified operator.

Fields

Name

Specify the name of the attribute to test.

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.
Not Available	Available would return False.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.

Mode	Description
Destination DN	Compares 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
- Not Equal

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.
Not Available	Available would return False.
Equal	There is a destination DN available, and it equals the specified value when compared using semantics appropriate to the DN format of the destination data store.
Not Equal	Equal would return False.
In Container	There is a destination DN available, and it represents an object in the container, specified by value, when compared using semantics appropriate to the DN format of the destination data store.
Not in Container	In Container would return False.
In Subtree	There is a destination DN available, and it represents an object in the subtree, specified by value, when compared using semantics appropriate to the DN format of the destination data store.
Not in Subtree	In Subtree would return False.

Value

Contains the value defined for the select operator. The value is used by the condition.

- Equal
- Not Equal
- In Continer
- Not in Container
- In Subtree
- Not in Subtree

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.

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.
Not available	Available would return False.
Equal	There is a value available for the specified attribute in the destination data store that equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
Changing	The current operation contains a change (modify attribute or add attribute) of the named entitlement.
Not Changing	Changing would return False.
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 using the specified comparison mode.
Not Changing From	Changing From would return False.
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 using the specified comparison mode.
Not Changing To	Changing To would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal
- Changing To
- Changing From
- Not Changing To
- Not Changing From

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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
- Changing To
- Changing From
- Not Changing To
- Not Changing From

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 Policies Components (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policycomponents.html).

Fields

Name

Specify the name of the global value to test for the selected condition.

Operator

Select the condition test type.

Operator	Returns True when...
Available	There is a global configuration value with the specified name.
Not Available	Available would return False.
Equal	There is a global configuration value with the specified name, and its value equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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.

Mode	Description
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

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 Policies Components (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policycomponents.html).

Fields

Name

Specify the name of the local variable to test for the selected condition.

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.
Not Available	Available would return False.
Equal	There is a local variable with the specified name, and its value equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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.

Mode	Description
Regular Expression	The regular expression matches the entire string. It defaults to case insensitive, but can be changed by an escape in the expression. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

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.

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.

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.

Operator

Select the condition test type.

Operator Returns True when...

Operator	Returns True when...
Available	There is a value available in the current operation other than a remove value for the specified attribute.
Not Available	Available would return False.
Equal	There is a value available in the current operation other than a remove value for the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.
Changing	The current operation contains a change other than a remove value for the specified attribute.
Not Changing	Changing would return False.
Changing From	The current operation contains a change that removes a value other than a remove value of the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Changing From	Changing From would return False.
Changing To	The current operation contains a change that adds a value other than a remove value to the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Changing To	Changing To would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal
- Changing To
- Changing From
- Not Changing To

- Not Changing From

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. See Sun’s Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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
- Not Equal
- Changing To
- Changing From
- Not Changing To
- Not Changing From

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 and 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.

Operator

Select the condition test type.

Operator	Returns True when...
Available	There is an operation property with the specified name on the current operation.
Not Available	Available would return False.
Equal	There is a an operation property with the specified name on the current operation, and its value equals the provided content when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.

Mode	Description
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

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 using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

The values are the operations that the Metadirectory engine looks for:

- add
- add-association
- check-object-password
- check-password
- delete
- get-named-password
- init-params
- instance
- modify
- modify-association
- modify-password
- move
- password
- query
- query-schema
- remove-association
- rename
- schema-def
- status

- sync

This list is not exclusive. Custom operations can be implemented by drivers and administrators.

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.
Destination DN	Compares 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

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.
Not Available	Available would return False.

If Source Attribute

Performs a test on attribute values of the current object 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.

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.
Not Available	Available would return False.
Equal	There is a value available in the source data store for the specified attribute. It equals the specified value when compared using the specified comparison mode.
Not Equal	Equal would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal

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. See Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html). The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.
Source DN	Compares using semantics appropriate to the DN format for the source data store.

Mode	Description
Destination DN	Compares 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
- Not Equal

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.
Not Available	Available would return False.
Equal	There is a source DN available, and it equals the content of the specified value in-container.
Not Equal	Equal would return False.
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 using semantics appropriate to the DN format of the source data store.
Not In Container	In Container would return False.
In Subtree	There is a source DN available, and it represents an object in the subtree identified by the specified value.
Not In subtree	In Subtree would return False.

Value

Contains the value defined for the select operator. The value is used by the condition. The operators that contain the value field are:

- Equal
- Not Equal
- In Container
- Not in Container
- In Subtree
- Not in Subtree

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.

Remarks

For more information on using XPath expression with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Pre-Identity Manager 3.5 Actions 18

Actions are performed when conditions of the enclosing rule are met. Some actions have a *Mode* field. The mode is not honored at run time if the context in which the policy is running is incompatible with the selected mode.

This section contains detailed information about all actions that are available through using the pre-Identity Manager Policy Builder interface.

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.

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.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

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.

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.

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.

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.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

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.

Value

Specify the attribute value to be added.

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.

DN

Specify the DN of the object to be added.

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.

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.

XPath Expression

Specify an XPath 1.0 expression that returns a node set containing the elements to which the new elements should be appended.

Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Append XML Text

Appends the specified text to the set of elements selected by the XPath expression.

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.

String

Specify the text to be appended.

Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Break

Ends processing of the current operation by the current policy.

Fields

There are no fields for the Break action.

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.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

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 the current object as the target object.

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.

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.

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 mapping purposes if the object is other than current object.

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.

Clear SSO Credential

Clears the Single Sign On credential so objects can be deprovisioned. Additional information about the credential to be cleared can be entered 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 Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object.

Target User DN

Specify the DN of the target users.

Application Credential ID

Specify the application credential that is stored in the application object.

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Clone By XPath Expression

Appends deep copies of the nodes specified by the source field to the set of elements specified by the destination field.

Fields

Source XPath Expression

Specify the XPath 1.0 expression that returns a node set containing the nodes to be copied.

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.

Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

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.

Destination Name

Specify the name of the attribute to be copied to.

Delete Destination Object

Deletes an object in the destination data store.

Fields

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.

Delete Source Object

Deletes an object in the source data store.

Fields

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.

Find Matching Object

Finds a match for the current object in the destination data store.

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.

Remarks

Find Matching Object is only valid when the current operation is an add.

The DN argument is required when scope is “entry,” and is optional otherwise. At least one match attribute is required when scope is “subtree” or “subordinates.”

The results are undefined if 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 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 &#xFFFC;. If multiple results are returned, then the destination DN of the current operation is set to the single character &#xFFFD;.

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.

Generate Event

Sends a user-defined event to Novell Audit or Sentinel.

Fields

ID

ID of the event. The provided value must result in an integer in the range of 1000-1999 when parsed using the parseInt method of java.lang.Integer.

Level

Level of the event.

Level	Description
log-emergency	Events that cause the Metadirectory engine or driver to shut down.
log-alert	Events that require immediate attention.
log-critical	Events that can cause parts of the Metadirectory engine or driver to malfunction.
log-error	Events describing errors that can be handled by the Metadirectory engine or driver.
log-warning	Negative events not representing a problem.
log-notice	Events (positive or negative) that an administrator can use to understand or improve use and operation.
log-info	Positive events of any importance.
log-debug	Events of relevance for support or engineers to debug the operation of the Metadirectory engine or driver.

Strings

Specify user-defined string, integer, and binary values to include with the event. These values are provided using the Named String Builder.

Tag	Description
target	The object being acted upon.
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
subTarget	The subcomponent of the target being acted upon.
text1	Text entered here is stored in the text1 event field.

Tag	Description
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.
value3	Any number entered here is stored in the value3 event field.
data	Data entered here is stored in the blob event field.

Remarks

The Novell Audit or 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.

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.

Move Destination Object

Moves an object into the destination data store.

Fields

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.

Move Source Object

Moves an object in the source data store.

Fields

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.

Reformat Operation Attribute Value

Reformats all values of an attribute within the current operation by using a pattern.

Fields

Name

Specify the name of the attribute.

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.

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.

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.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

Select 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 syntax of the new attribute value.

String

Specify the value of the new attribute.

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.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

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 syntax of the attribute value to be removed.

String

Specify the attribute value to be removed.

Rename Destination Object

Renames an object in the destination data store.

Fields

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.

String

Specify the new name of the object.

Rename Operation Attribute

Renames all occurrences of an attribute within the current operation.

Fields

Source Name

Specify the original attribute name.

Destination Name

Specify the new attribute name.

Rename Source Object

Renames an object in the source data store.

Fields

Select Object

Select the target object. This object can be the current object, or can be specified by a DN or an association.

String

Specify the new name of the object.

Send Email

Sends an e-mail notification.

Fields

ID

(Optional) Specify the User ID in the SMTP system sending the message.

Server

Specify the SMTP server name.

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 Using Named Password in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

Message Type

Select the e-mail message type.

Strings

Specify the values containing the various e-mail addresses, subject, and message. The following table lists valid named string arguments:

String Name	Description
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.
from	Specifies the address to be used as the originating e-mail address.
reply-to	Specifies the address to be used as the e-mail message reply address.
subject	Specifies the e-mail subject.
message	Specifies the content of the e-mail message.
encoding	Specifies the character encoding to use for the e-mail message.

Send Email from Template

Generates an e-mail notification using a template.

Fields

Notification DN

Specify the slash form DN of the SMTP notification configuration object.

Template DN

Specify the slash form DN of the e-mail template object.

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 Using Named Passwords in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

Strings

Specify additional fields for the e-mail message. The following table contains reserved field names, which specify the various e-mail addresses:

String Name	Description
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.
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.

Each template can also define fields that can be replaced in the subject and body of the e-mail message.

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.

Write Back

Select whether or not to also write back the default values to the source data store.

Values

Specify the default values of the attribute.

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.

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.

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.

Set Destination Password

Sets the password for an object in the destination data store.

Fields

Mode

Select whether this action should be added to, before, or after the current operation, or written directly to the destination data store.

String

Specify the password to be set.

Set Local Variable

Sets a local variable with the given name to the string value specified, the XPath 1.0 Node Set specified, or the Java* Object specified.

Fields

Variable Name

Specify the name of the new local variable.

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 value of the variable, in the format required by the *Variable Type* field.

Set Operation Association

Sets the association value for the current operation.

Fields

Association

Specify the new association value.

Set Operation Class Name

Sets the object class name for the current operation.

Fields

String

Specify the new class name.

Set Operation Destination DN

Sets the destination DN for the current operation.

Fields

DN

Specify the new destination 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.

String

Specify the name of the string.

Set Operation Source DN

Sets the source DN for the current operation.

Fields

DN

Specify the new source DN.

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.

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.

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.

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.

Set Source Password

Sets the password for an object in the source data store.

Fields

String

Specify the password to be set.

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 Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object.

Target User DN

Specify the DN of the target users.

Application Credential ID

Specify the application credential that is stored in the application object.

Login Parameter Strings

Specify each login parameter for the application. The login parameters are the authentication keys stored in the application object.

Set SSO Passphrase

Sets the Novell SecureLogin passphrase and answer when a User object is provisioned. This action is part of the Credential Provisioning policies. For more information, see Novell Credential Provisioning Policies (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy_credprov/data/bookinfo.html).

Fields

Credential Repository Object DN

Specify the DN of the repository object.

Target User DN

Specify the DN of the target users.

Question Strings

Specify the SecureLogin passphrase question.

Answer String

Specify the SecureLogin passphrase answer.

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.

XPath Expression

XPath 1.0 expression that returns a node set containing the elements on which the XML attribute should be set.

String

Specify the value of the XML attribute.

Remarks

For more information on using XPath expressions with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Status

Generates a status notification.

Fields

Level

Specify the status level of the notification. The levels are error, fatal, retry, success, and warning.

Message

Provide the status message 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.

Strip Operation Attribute

Strips all occurrences of an attribute from the current operation.

Fields

Name

Specify the name of the attribute to be stripped.

Strip XPath

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.

Remarks

For more information on using XPath expression with policies, see XPath 1.0 Expressions (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

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 Versioning Information in the Novell Identity Manager Administration Guide (http://www.novell.com/documentation/idm35/index.html).

Color

Select the color of the trace message.

String

Specify the value of the trace message.

Veto

Vetoes the current operation.

Fields

There are no fileds.

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.

Pre-Identity Manager 3.5 Noun Tokens 19

Noun tokens expand to values that are derived from the current operation, the source or destination data stores, or some external source.

This section contains detailed information about all noun tokens that are available through using the pre-Identity Manager Policy Builder interface.

Added Entitlement

Expands to the values of an entitlement granted in the current operation.

Fields

Name

Name of the entitlement.

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.

Association

Expands to the association value from the current operation.

Fields

There are no fields.

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.

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.

Class Name

Expands to the object class name from the current operation.

Fields

There are no fields.

Destination Attribute

Expands to the specified attribute value an object.

Fields

Name

Name of the attribute.

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

Select Object

Select Current Object, DN, or 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.

Destination DN

Expands to the destination DN specified in the current operation.

Fields

Convert

Select whether or not to convert the DN to the format used by the source data store.

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.

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.

Destination Name

Expands to the unqualified Relative Distinguished Name (RDN) of the destination DN specified in the current operation.

Fields

There are no fields.

Entitlement

Expands to the values of a granted entitlement from the current object.

Fields

Name

Name of the entitlement.

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.

Global Configuration Value

Expands to the value of a global configuration variable.

Fields

Name

Name of the global configuration value.

Local Variable

Expands to the value of a local variable.

Fields

Name

Specify the name of the local variable.

Named Password

Expands to the Named Password from the driver.

Fields

Name

Specify the Named Password.

Operation

Expands to the name of the current operation.

Fields

There are no fields.

Operation Attribute

Expands to the value of an attribute from the current operation. It does not include the removed values from a modify operation.

Fields

Name

Specify the name of the 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.

Password

Expands to the password specified in the current operation.

Fields

There are no fields.

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 to remove.

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.

Removed Entitlements

Expands to the values of the an entitlement revoked in the current operation.

Fields

Name

Specify the name of the entitlement.

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.

Source Attribute

Expands to the values of an attribute from an object in the source data store.

Fields

Class Name

(Optional) Specify the class name of the target object. Leave the field blank to use the class name from the current object.

Name

Name of the attribute.

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.

Source DN

Expands to the source DN from the current operation.

Fields

Convert

Select whether or not to convert the DN to the format used by the destination data store.

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.

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.

Source Name

Expands to the unqualified relative distinguished name (RDN) of the source DN specified in the current operation.

Fields

There are no fields.

Text

Expands to the text.

Fields

Text

Specify the text.

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.

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 be specified by a DN or association.

Pattern

Specify patterns to use to generate unique values by using the Argument Builder.

Counter Start

The starting value of the counter.

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.

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 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.

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.

XPath

Expands to 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 (http://www.novell.com/documentation/idm35/index.html?page=/documentation/idm35/policy/data/policyxpathexpression.html#policyxpathexpression).

Pre-Identity Manager 3.5 Verb Tokens

20

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 pre-Identity Manager Policy Builder interface.

Escape Destination DN

Escapes the enclosed tokens according to the rules of the DN format of the destination data store.

Fields

There are no fields.

Escape Source DN

Escapes the enclosed tokens according to the rules of the DN format of the source data store.

Fields

There are no fields.

Lowercase

Converts the characters in the enclosed tokens to lowercase.

Fields

There are no fields.

Parse DN

Converts the enclosed token's DN to an alternate format.

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 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.

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.

Replace With

Specify the replacement string.

Remarks

For details on creating regular expressions, see:

- Sun's Java Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html)
- Sun's Java Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Matcher.html#replaceAll (java.lang.String))

The pattern options CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed by using the appropriate embedded escapes.

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.

Replace With

Specify the replacement string.

Remarks

The matching instance is replaced by the string specified in the *Replace with field*.

For details on creating regular expressions, see:

- Sun's Web site (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html)
- Sun's Web site (java.lang.String) (http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Matcher.html#replaceAll (java.lang.String))

The pattern option CASE_INSENSITIVE, DOTALL, and UNICODE_CASE are used but can be reversed using the appropriate embedded escapes.

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 the first character from the end. If -3 is specified, then is 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 or the original string. If -2 is specified, the length is the entire -1. For a string with 5 characters a length of -1 = (5 + (-1)) + 1 = 5, -2 = (5 + (-2)) + 1 = 4, etc.

Uppercase

Converts the characters in the enclosed tokens to uppercase.

Fields

There are no fields.